Viz Mosart Administrator Guide

Version 5.18

Viz Mosart

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

2026/10/12

Contents

1 Introduction 14
1.1 Related Documents 14
1.2 Feedback and Suggestions 14
2 About Viz Mosart 15
2.1 Naming Convention for Components 16
2.2 The Viz Mosart System 18
2.2.1 Viz Mosart Applications 19
2.3 Viz Mosart Server 20
2.3.1 Viz Mosart Server 20
2.3.2 Viz Mosart Server Components 21
2.4 Other Viz Mosart Applications 31
2.4.1 Media Router and Mosart Template Database 31
2.4.2 Omnibus Controller 31
2.4.3 SNMP Service 31
2.4.4 Test Suite 31
2.5 Viz Mosart Clients 33
2.5.1 Viz Mosart GUI 33
2.5.2 NRCS Plugin 34
2.5.3 Timing Display 35
2.5.4 Rundown Viewer 35
2.5.5 iNEWS Timer 36
2.5.6 Audio Panel 37
2.5.7 Audio Player 38
3 Installation 39
3.1 Prerequisites 40
3.1.1 System Requirements 40
3.1.2 Hardware Requirements 42
3.1.3 Licensed and Optional Features 45
3.2 Redundancy 46
3.2.1 Automation Server Mirroring 48
3.2.2 Connected GUIs 48
3.2.3 Graphical Interfaces 48
3.2.4 Device-specific Redundancy 49
3.3 Licensing 51

3.3.1 License Management 51
3.3.2 License Configuration status 51
3.3.3 License Notifications 52
3.4 Viz Mosart Installation Files 53
3.4.1 Viz Mosart Installer Files 53
3.4.2 Additional Mosart Files 54
3.4.3 Installation and Upgrade 54
3.4.4 Version Numbering 57
4 Security in Viz Mosart Ecosystem 58
4.1 Introduction 58
4.2 What is secured, and by what 58
4.3 How the layers fit together 59
4.4 Roles across Viz Mosart 59
4.5 Setup at a glance 60
4.6 End to end verification 61
4.7 Troubleshooting 61
4.8 Choosing an Authentication Mode 64
4.8.1 Do you need the Mosart Web Apps? 64
4.8.2 Third-party integrations and the REST API 65
4.8.3 What works when OIDC is enabled 66
4.9 Identity Provider and Certificates 67
4.9.1 The identity provider 67
4.9.2 The scopes 68
4.9.3 The roles claim 68
4.9.4 The certificate 68
4.9.5 Trust the certificate everywhere 69
4.10 Securing the Mosart Server 70
4.10.1 Turn on HTTPS 70
4.10.2 Turn on OIDC authentication 70
4.10.3 Roles the Mosart server enforces 71
4.10.4 Check the server on its own 71
4.11 Password-Protected Features 73
4.11.1 Template Editor password 73
4.11.2 GPO 73
4.11.3 Generic REST driver credentials 73
4.11.4 Vizrt Graphics MSE database 73
4.11.5 TriCaster 73

5 AV Automation 74
5.1 Overview 74
5.1.1 AV Automation Tasks 74
5.1.2 AV Automation Characteristics 75
5.2 Menu Selections 76
5.3 Logging and Status 77
5.3.1 Logging Panel 77
5.4 AV Automation Device Properties 79
5.4.1 Vision Mixer 81
5.4.2 Video Servers 90
5.4.3 Graphics 120
5.4.4 Vizrt Graphics 129
5.4.5 Audio 130
5.4.6 Camera Robotics 151
5.4.7 Light Control 155
5.4.8 Router 157
5.4.9 Subtitles 158
5.4.10 GPI/O 159
5.4.11 Tally 164
5.4.12 Virtual Set 167
5.4.13 Weather 168
5.4.14 Video Wall 169
5.4.15 General 170
5.5 A/V Setup (Audio and Video Setup Config File) 172
5.5.1 Working with Audio and Video Setup 172
5.6 Device Connection Strings and Configuration Files 181
5.6.1 Video Wall Connection Strings 182
5.6.2 Device Configuration Files 185
5.6.3 Video Server, MAM and Database Connection Strings 216
5.7 Template Editor 269
5.8 Mosart Instrumentation Panel 270
5.9 Recording a Show 271
5.9.1 Working with Recordings 271
5.9.2 Supported Video Servers 273
6 Manus Administrator 274
6.1 NRCS Software Compatibility 274
6.1.1 Installation 274

6.2 Configuration Utilities....275
6.2.1 Centralized Settings Menu....275
6.2.2 Newsroom Settings....276
6.3 Settings Editor iNews....277
6.3.1 Working with iNEWS Settings....277
6.4 Settings Editor MOS....287
6.4.1 Working with MOS Settings....287
6.4.2 Notes....298
6.5 Configuring the MSE Playlist Panel....299
6.5.1 Managing the MSE Playlist Panel....299
6.6 Field Mapping....300
6.6.1 The Tool Set....300
6.6.2 What to Configure....300
6.6.3 Target Fields....300
6.6.4 Mapping Source Fields to Target Fields....301
6.6.5 Examples....302
6.7 Manus Admin Commands....308
6.8 Newsroom Settings Editor....310
6.8.1 Working with the Newsroom settings....311
6.8.2 Viz Mosart Object Structure....319
6.8.3 Viz Mosart and Viz Content Pilot (VCP) MOS Item Integration....323
6.8.4 MOS ID Mapping....327
6.8.5 Additional Newsroom Settings Examples....328
6.9 iNEWS Timer....336
6.9.1 Working with the iNEWS Timer....336
7 Media Administrator....339
7.1 Media Administrator Properties Editor....340
7.1.1 Sections of the Mosart Media Administrator Properties File....342
7.1.2 Dynamic Video Server Configuration....344
7.2 Media Administrator Commands....346
8 Overlay Graphics Interface....348
8.1 Overlay Graphics Interface Overview....349
8.1.1 [1] List of Overlay Graphics in the Rundown....349
8.1.2 [2] Server Status....351
8.1.3 [3] General Log....352
8.1.4 [4] Event Log....352
8.1.5 System Information....352

8.1.6 Menu Bar 352
8.2 Overlay Graphics Configuration and Use 354
8.3 Mosart Graphics Destinations and Takeout Logic 355
8.3.1 Mosart Graphics Destinations 355
8.3.2 Takeout Logic 359
8.4 Sending Lower Third Graphics Details to an NRCS 362
8.4.1 Managing Destinations 362
8.4.2 Detailed Description of MOS Objects 363
8.5 Overlay Graphics Configuration Property Tabs 364
8.5.1 Working with Overlay Graphics Configurations 364
8.6 Overlay Graphics Types 370
8.6.1 VIZRT 370
8.6.2 VIZ FLOWICS 372
8.6.3 CHYRON 374
8.6.4 DEKO 374
8.6.5 ORAD 375
8.6.6 PIXEL POWER CC 376
8.6.7 XPRESSION 377
8.6.8 BIGVIZ 377
8.7 Named Overlay Graphics 378
8.7.1 Working with Named Overlay Graphics 378
8.7.2 NRCS-based Systems 379
8.7.3 Viz Flowics-based Systems 382
8.8 Managing Overlay Graphics Controllers and Engines 384
8.8.1 Overlay Graphics Controllers and Engines 384
9 Show Design and Template Editor 390
9.1 Overview 391
9.1.1 Template Sets 391
9.1.2 Working with Templates 392
9.1.3 Template Terms and Definitions 393
9.2 Template Examples 395
9.2.1 Studio Camera 395
9.2.2 Package: Video Clip with Full Sound 396
9.2.3 Voice-over 397
9.2.4 Live External Source 398
9.2.5 DVE 399
9.2.6 Graphic: Full Screen Graphics 403

9.2.7 Accessory Templates....403
9.2.8 Switcher Plugin....405
9.3 Working with Templates....414
9.3.1 Template Operations....415
9.3.2 Template Properties....417
9.3.3 Template Device Functions....422
9.3.4 Newsroom Tags....472
9.3.5 Control Commands in Templates....484
9.3.6 Destination Panel....512
9.3.7 Empty Templates....513
9.3.8 Configuring a REST-based Driver....521
9.3.9 Mosart Templates Feedback to the NRCS....538
9.4 Other Template Functionality....546
9.4.1 Additional Template Features....546
9.5 Working with Template Sets....550
9.5.1 Template Set Types....550
9.5.2 Template Set Operations....552
9.5.3 Template Set Properties....555
9.5.4 Template Set Statistics....557
10 Viz Mosart Central Settings....559
10.1 Working with the Settings Menu....559
10.1.1 Opening the Settings Menu....560
10.1.2 Architecture....560
10.1.3 Setup Examples....560
10.2 Vizrt Graphic Settings....561
10.2.1 To configure Vizrt graphics....561
10.3 Generic REST Drivers....564
10.3.1 Setting up and Connecting to a REST-based Device....564
10.3.2 Authentication types....566
10.3.3 Management of REST-based Devices....567
10.3.4 Controlling a REST-based Driver....568
10.3.5 Troubleshooting....568
10.3.6 REST Driver Implementation....569
11 Frame Accurate System Operations....572
11.1 Why Frame Accuracy Matters....572
11.1.1 Predictable Behavior of Equipment....572
11.1.2 Latency Compensation....573

11.2 Requirements 573
11.2.1 Choosing a clock source 574
11.2.2 Plura card models and firmware 574
11.3 Enabling Frame Accuracy Operation 575
11.3.1 Configuring Viz Mosart for Frame Accurate Operation 575
11.3.2 Genlock and Timecode Reliability 577
11.4 UTC or Local Time 577
11.5 Calibrating the System 578
11.5.1 Timecode Readers 579
11.5.2 Gathering Clock Source Statistics for Vizrt Support 580
11.6 Genlock Logging 582
11.7 Frame Accuracy in Story Recorder 582
11.8 Limitations 582
11.9 Device Calibration 583
11.9.1 Device Calibration 583
11.9.2 General setup 584
11.9.3 Calibration Overview 587
11.9.4 Before you Start 588
11.9.5 Test Clips 589
11.9.6 Rundown Recipes 590
11.9.7 A-series Calibration Tests: The Recommended Set 590
11.9.8 B-series calibration tests: alternative tests 602
11.10 Show Design for Frame Accuracy and Story Recorder 604
11.10.1 Introduction 604
11.10.2 Background 605
11.10.3 Definitions 605
11.10.4 Show Design Table Columns 607
11.10.5 Show Design Table 608
11.11 PTP Setup 626
11.11.1 PTP Setup 626
11.11.2 Requirements 626
11.11.3 Supported Deployment Scenarios 627
11.11.4 Setting up the Windows PTP Provider 628
11.11.5 Viz Mosart Configuration 631
11.11.6 Verification 633
11.11.7 Troubleshooting 635
12 Operational Examples 638

12.1 Device-Specific Operations 639
12.1.1 Viz Arc support within a Mosart show 640
12.1.2 Viz Trio support in a Viz Mosart Show....648
12.1.3 Viz Engine as a Switcher....654
12.1.4 Avid iNews Web Service for Status Feedback 693
12.1.5 Jupiter Systems PixelNet Video Wall Control 719
12.1.6 Panasonic Kairos Vision Mixer 724
12.1.7 Stagetec Driver Configuration 732
12.1.8 Stream Deck 738
12.1.9 TriCaster and Viz Mosart 739
12.1.10 Adaptive Live Producer....749
12.2 Media Sequencer....803
12.2.1 Configuring Vizrt Graphics 803
12.2.2 Main/Backup Topology 804
12.3 Mosart Remote Control REST API....807
12.3.1 Getting started....807
12.3.2 Endpoints....809
12.4 Story Recorder 833
12.4.1 Introduction to Story Recorder 833
12.4.2 Story Recorder Highlights....834
12.4.3 Definitions....834
12.4.4 Story Recorder Setup 837
12.4.5 Working with Story Recorder....893
12.4.6 Story Recorder Troubleshooting 915
13 Viz Mosart Template Database....937
13.1 Architecture....937
13.2 Template Properties 937
13.3 Information Stored in the Template Database 938
13.4 Prerequisites for Connecting to TDB....938
13.5 Creating and Upgrading Mosart templatedb....940
13.5.1 Upgrading MySQL 5.x to MySQL 8.x....940
13.5.2 Verifying the WampServer 940
13.5.3 Running Database Scripts....941
13.5.4 Running Scripts Automatically 941
13.5.5 Running Scripts Manually 942
13.6 Backup and Recovery 943
13.6.1 Working with the Server Database 943

13.6.2 Working with Database Clients 944
13.7 Database Maintenance 946
13.7.1 Upgrading the Database 946
13.7.2 Viz Mosart Template Editor 947
13.7.3 TemplateSetEditor 947
13.8 Viz Mosart Template Database Specification 956
13.8.1 Entity Relationships 956
13.8.2 Introductions and Notations 957
13.8.3 Viz Mosart Database Tables 960
13.9 Configuring the Viz Mosart Server 979
13.9.1 AV Automation and the Mosart Template Database 979
13.9.2 Manus Administrator and the Mosart Template Database 985
13.9.3 Mosart Web Apps NRCS Plugin 986
13.9.4 Viz Mosart ActiveX 987
13.10 Installing the Viz Mosart Database 989
13.10.1 MySQL Database Servers and Clients 989
13.10.2 MySQL Users 990
13.10.3 Initial Mosart Template Database 990
14 Maintenance 992
14.1 General Advice on System Operations 993
14.1.1 Rebooting and Restarting - General Notes 993
14.1.2 Rebooting Machines 993
14.1.3 Application Restart 993
14.1.4 Viz Mosart Application Updates 993
14.1.5 Windows Updates 993
14.1.6 Viz Mosart Main/Backup Server Testing 994
14.1.7 AV Automation Slow Start 994
14.2 System Logging 995
14.2.1 Log Viewer 995
14.2.2 Log File Characteristics 996
14.2.3 Log Content 997
14.2.4 Log Adapters 999
14.2.5 Configuring Logging 1000
14.2.6 Starting-up a Log Adapter 1005
14.2.7 Troubleshooting Log Adapters 1005
14.3 Server Maintenance 1008
14.3.1 Server File Structure 1008

14.3.2 Files for Backup 1010
14.3.3 File Purging 1010
15 Appendix 1011
15.1 GPIO with Sample XML Config 1012
15.1.1 Sample Configuration (W&T Web-IO) 1012
15.1.2 Hardware Configuration 1060
15.2 Structure of MOS Objects Sent to NRCS 1062
15.2.1 MOS Object with Mosart Templates Grouped by Type 1063
15.2.2 MOS Object with Mosart Templates not Grouped by Type 1071
15.2.3 MOS Object with Clip Templates Merged in a Mosart Template 1074
15.2.4 Structure of the MOS object with Lowerthird Information 1077
15.2.5 Octopus 1080
15.3 Vizrt SSO (Keycloak) 1081

This manual is organized as follows:

- Introduction
- About Viz Mosart
- Installation
- Security in Viz Mosart Ecosystem
- AV Automation
- Manus Administrator
- Media Administrator
- Overlay Graphics Interface
- Show Design and Template Editor
- Viz Mosart Central Settings
- Frame Accurate System Operations
- Operational Examples
- Viz Mosart Template Database
- Maintenance
- Appendix

Copyright © 2026 Vizrt

1 Introduction

This guide takes you through the setup of Viz Mosart and is a reference guide for use during its installation and configuration. The purpose of this document is to help new users become familiar with the system; to illustrate the main workflow, and to show the available options. Familiarity with XML is required.

1.1 Related Documents

- *Viz Mosart User Guide*: Contains information on how to use Viz Mosart in live production.

For more information about all of the Vizrt products, visit:

- www.vizrt.com
- Vizrt Documentation Center
- Viz University
- Vizrt Forum

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

2 About Viz Mosart

Viz Mosart is a broadcast solution designed for newsroom environments. Mosart allows a control room to be fully operational with minimal effort and consistency within the production.

Viz Mosart is a collection of server and client applications, along with services, that control equipment within your broadcast environment.

This section contains the following topics:

- Naming Convention for Components
- The Viz Mosart System
- Viz Mosart Server
- Other Viz Mosart Applications
- Viz Mosart Clients

2.1 Naming Convention for Components

When contacting support, please use the following standard naming convention for Viz Mosart applications and components:

Short Name	Long Name	Description
AVA	AV Automation	Application controlling attached broadcast equipment.
GUI	Viz Mosart GUI	Main user control application.
KVM	Keyboard, Video, Mouse	Application allowing multiple computers to share the same monitor, keyboard and mouse.
Manus Admin	Manus Administrator	Application controlling the Viz Mosart Rundown.
Media Admin	Media Administrator Configuration	Media Administrator. Application for monitoring media objects (clips).
MMR	Media Router	Application that shares broadcast devices between control rooms. See Media Router and Mosart Template Database.
NCS, NRCS	Newsroom Computer System	
OGI	Overlay Graphics Interface	Application that controls Overlay graphics systems.
RCC	Robotic Camera Controller	
RCS	Remote Control Service	Service used for all external PCs, like GUI and Timing Display, to connect to Viz Mosart Server. Sometimes referred to as RCPS.
Trio Interface		Application that controls Vizrt overlay graphics. (Kept for backward compatibility only - use Overlay Graphics Interface instead).
URL	Uniform Resource Locator	In this context, address of web service.

Copyright © 2026 Vizrt

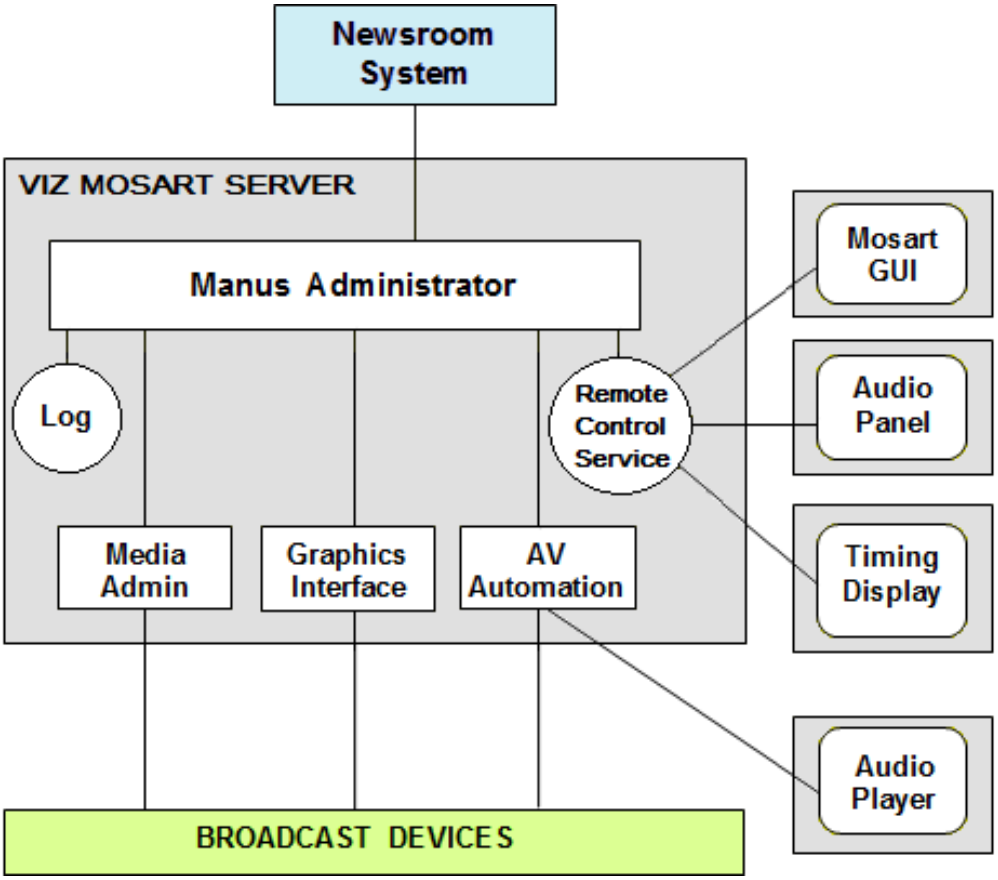
Short Name	Long Name	Description
Viz Mosart	Viz Mosart Newsroom Automation System	

2.2 The Viz Mosart System

The Viz Mosart system is a collection of software applications that join to make your production come together. The applications used may vary between installations and are dependent on your broadcast environment.

The following is a general and simplified overview of the Viz Mosart system.

Viz Mosart comprises two main applications: the **Viz Mosart Server** and the **Viz Mosart GUI**. There are also several supporting applications, such as the **Timing Display** and **Audio Panel**. You can have more than one Viz Mosart GUI running at workstations connected to the Mosart server. Normally there are two Mosart servers: one *active* (live) and one *backup* server.



Note: The shaded boxes in the diagram signify a physical device such as a workstation, server or a piece of hardware. White rectangles represent Viz Mosart applications, white circles represent Viz Mosart services, and rounded rectangles represent user displays.

The figure above shows a typical Viz Mosart installation with one **Viz Mosart Server** and four Viz Mosart workstations to run the user control software. There is also provision for a physically attached remote audio fader panel attached to a Viz Mosart workstation via the **Audio Panel** application. Redundancy is not shown.

All Viz Mosart applications and services are connected via TCP/IP, allowing them to run on any computer in a common logical network.

Copyright © 2026 Vizrt

2.2.1 Viz Mosart Applications

The Viz Mosart software package consists of the following applications:

Viz Mosart Server

- Viz Mosart Server
  - Manus Administrator Configuration
  - Media Administrator Configuration
  - Overlay Graphics Interface
  - AV Automation
  - Log Service and Log Viewer
  - Remote Control Service

Viz Mosart Client Applications

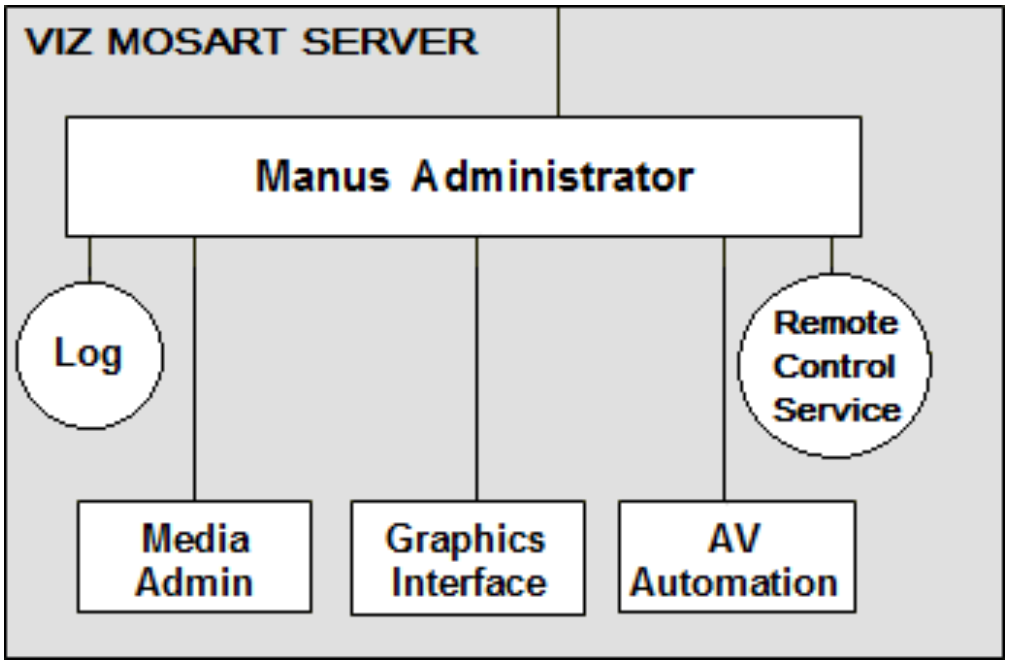
- Viz Mosart Clients
  - Viz Mosart GUI
  - Audio Panel (see *Viz Mosart User Guide*)
  - Timing Display (see *Viz Mosart User Guide*)
  - Audio Player
  - ActiveX Configuration
  - iNews Timer

Other Viz Mosart Applications

- Other Viz Mosart Applications
  - Media Router and Mosart Template Database
  - Omnibus Controller
  - SNMP Service
  - Test Suite

Copyright © 2026 Vizrt

2.3 Viz Mosart Server



2.3.1 Viz Mosart Server

The Viz Mosart server:

- Comprises a suite of dedicated server applications, relevant to your broadcast environment, running independently of one another on the same workstation.
- Supports multiple GUI workstations connected to it at any time. The control room has two Viz Mosart GUIs, one for redundancy, while others can be used for information purposes and locked to *Browse Mode* for monitoring the rundown outside the control room.
- Supports multiple timing displays. Various timing information can be seen at a glance wherever the application is installed. It may also be helpful to have a different display for the control room and another for the studio floor.
  The Timing Display is connected to the Viz Mosart rundown that is currently On Air.
  - Manus Administrator
  - Media Administrator
  - Overlay Graphics Interface
  - AV Automation
  - Log Service and Log Viewer
  - Remote Control Service

2.3.2 Viz Mosart Server Components

Icon	Application
	Mosart Manus Administrator Executables either: - (Default) *Vizrt.Mosart.ManusAdministrator.exe (*A combined iNEWS/MOS program). - *MMConsoleAdmin_2007.exe* (FTP, for iNEWS). - MMConsoleAdmin_MOS.exe (for MOS workflow Newsroom Control Systems).
	Mosart Media Administrator Executable: *MMMediaAdministrator.exe*
	Mosart AV Automation Executable: *MMAVAutomation.exe*
	Mosart Overlay Graphics Executable: *MMOverLayGraphicsInterface.exe*

The Viz Mosart server is split into four main components plus two services, each handling an important role for the system to function. Splitting the system into components allows greater flexibility in handling a variety of broadcast devices and productions.

A typical Viz Mosart server installation contains the following applications and services.

- Manus Administrator (console application): Handles the connection to the Newsroom Control System (NRCS).
- Media Administrator (console application): Application for monitoring media objects (clips) referenced in the current rundown. Supports the media search option used in the Viz Mosart GUI.
- Overlay Graphics Interface (Windows application): Application for controlling and monitoring overlay graphics.
- AV Automation (Windows application): Application for controlling all other external broadcast equipment.

Copyright © 2026 Vizrt

- Log Service and Log Viewer (Windows service): Service used for generating Viz Mosart command logs. Automatically installed with Viz Mosart Server and the Viz Mosart GUI.
- Remote Control Service (Windows service): Service used for all external PCs, like GUI and Timing Display, to connect to the Viz Mosart Server. Automatically installed with the Viz Mosart Server.

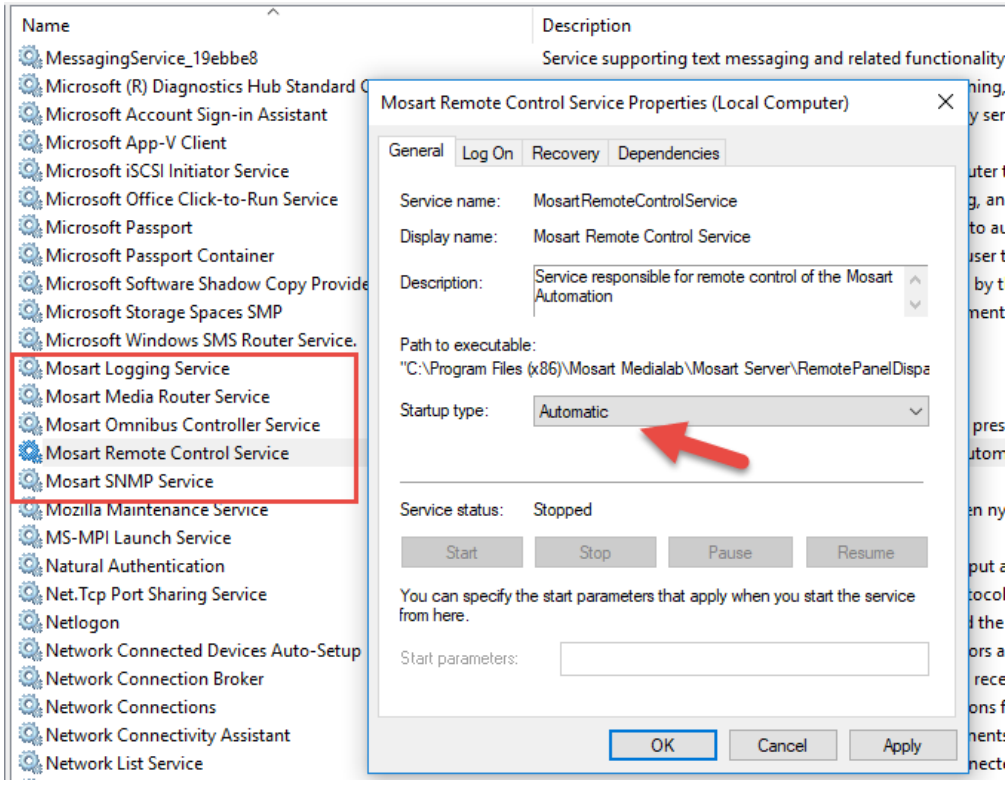
Normally, all of the services listed above should be running at all times on the Mosart server. To ensure this, you could start them from a convenience batch-script such as the one below:

StartMosartServicesAndGUI.cmd

```
@echo off

@REM Assume we are installed at the default location, otherwise fix path
cd "C:\Program Files (x86)\Mosart Medialab\Mosart Server"

start "Manus Admin" /MIN MMConsoleAdmin_MOS.exe
timeout /T 1 > nul
start "Media Admin" /MIN MMMediaAdministrator.exe
timeout /T 1 > nul
start "AvAutomation" /MIN MMAVAutomation.exe
timeout /T 2 > nul
start "OverlayGraphics" /MIN MMOverLayGraphicsInterface.exe
timeout /T 1 > nul
cd "C:\Program Files (x86)\Mosart Medialab\Mosart GUI"
start "Mosart GUI" MosartMultiGui.exe
```

The status of the console applications (Manus Admin and Media Admin) can easily be seen on the Viz Mosart server console. The same applies Windows applications **AvAutomation** and **OverlayGraphics**.

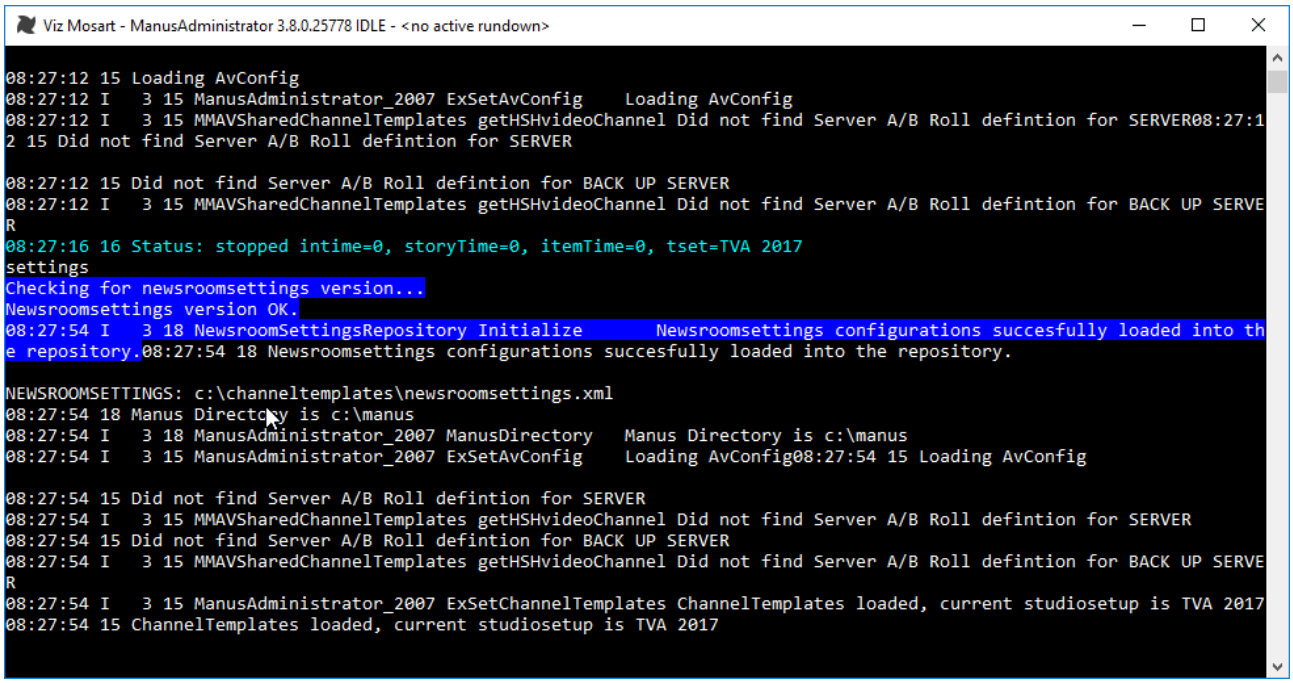
The **RCS (Remote Control Service)** and **Log Service** run as Windows services and are therefore not available for easy visual inspection. It's recommended to make these services start automatically when Windows starts up. One easy way to do this is to use the Windows services application (**Start button > Services**): locate the required services in the **services** window, right-click the service and make sure its properties are set to **Automatic** as shown in the screenshot below:



If you prefer the command line, you can easily inspect and start/stop services from Powershell. Start an elevated Powershell (Admin) session via **Start > Powershell** > right-click and select **Run as administrator**.
In the example below, the Mosart Logging Service and RCS are running, while other Viz Mosart services are stopped:

```
PS C:\Users\bva.VIZRTINT> Get-Service | grep Mosart
Running  MosartLogService   Mosart Logging Service
Stopped  MosartMediaRout... Mosart Media Router Service
Stopped  MosartOmnibusCo... Mosart Omnibus Controller Service
Running  MosartRemoteCon... Mosart Remote Control Service
Stopped  MosartSNMPService  Mosart SNMP Service
```

Manus Administrator



Manus is short for *manuscript*, the script originating from the Newsroom Control System (NRCS). Manus Administrator is the central Viz Mosart application that controls the current rundown in the Viz Mosart GUI and receives rundowns from the attached NRCS. It issues events according to the different story items in the rundown and is dynamically updated when changes are made in the NRCS. Although Manus Administrator receives information from the NRCS, it does not send updates back - this prevents rundown control conflicts between the systems.

- Manus Administrator handles the connection to the NRCS, and executes commands at the appropriate time according to GUI input and template configuration.
- Manus Administrator runs as a live console application and responds to free text commands. For improved execution speed, Manus Administrator should be run in the background or be minimized.

Variants

Manus Administrator has variants with slightly different settings that suit different broadcast environments. During installation, the newer *combined Manus* is installed by default.

- **Combined Manus** (default) for all NRCS connections. Installs the file *Vizrt.Mosart.ManusAdministrator*.

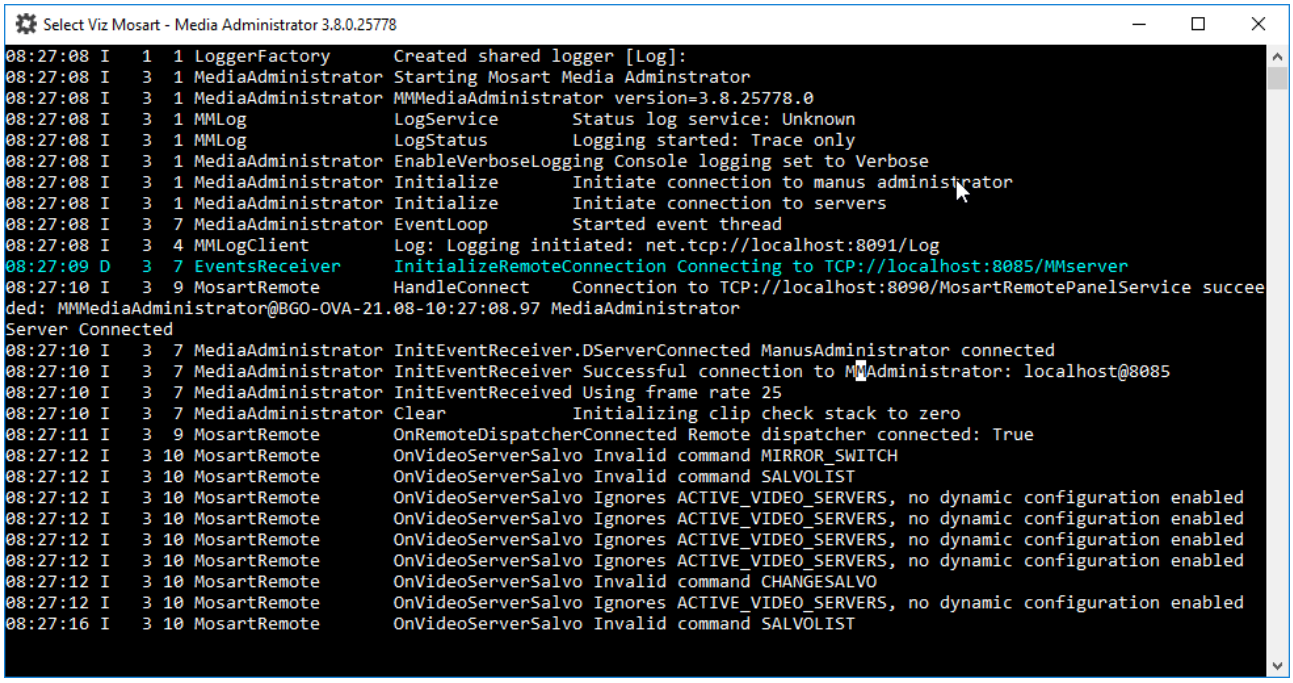
For non-standard setups, you can also manually select one of these variants from the Installation folder (*C:\Program Files (x86)\Mosart Medialab\Mosart Server*):

- **iNews FTP Newsroom** connections. Installs the file *MMConsoleAdmin_nnnn*.
- **MOS Newsroom** connections. Installs the file MMConsoleAdmin_MOS.

Copyright © 2026 Vizrt

> **Info:** Manus Administrator is sometimes also referred to as *MMConsoleAdmin*, *ManusAdmin*, *ManusAdministrator* or just *Manus*.

Media Administrator



The Media Administrator (MMMediaAdministrator) handles the database connection to your video servers. This application references the database to provide a current clip list within the Viz Mosart timeline, without making changes.

Mainly used for monitoring and searching video clips on connected video servers, the Media Administrator:

- Is responsible for reporting status for all clips in the current rundown to the Manus Administrator - typically whether a clip is present and properties like clip duration. The clip status for video servers is displayed in the Viz Mosart GUI as horizontal bars. A light-blue bar indicates a clip that is present on the video server, whilst a checkered bar indicates a non-existing clip. Clip length is shown by the length of the horizontal bar in the Viz Mosart GUI. For other media objects, Viz Mosart may present clip status in other ways.
- Makes it possible to **search** for clips on the video server. This functionality is used by the Viz Mosart **media search window**, making it possible to add clips to the rundown without the use of an NRCS. You can also search for other media objects like subtitles, graphic elements, and audio files.
  The Media Administrator runs as a live console application and responds to free-text commands. It should be run in the background or minimized for improved execution speed.

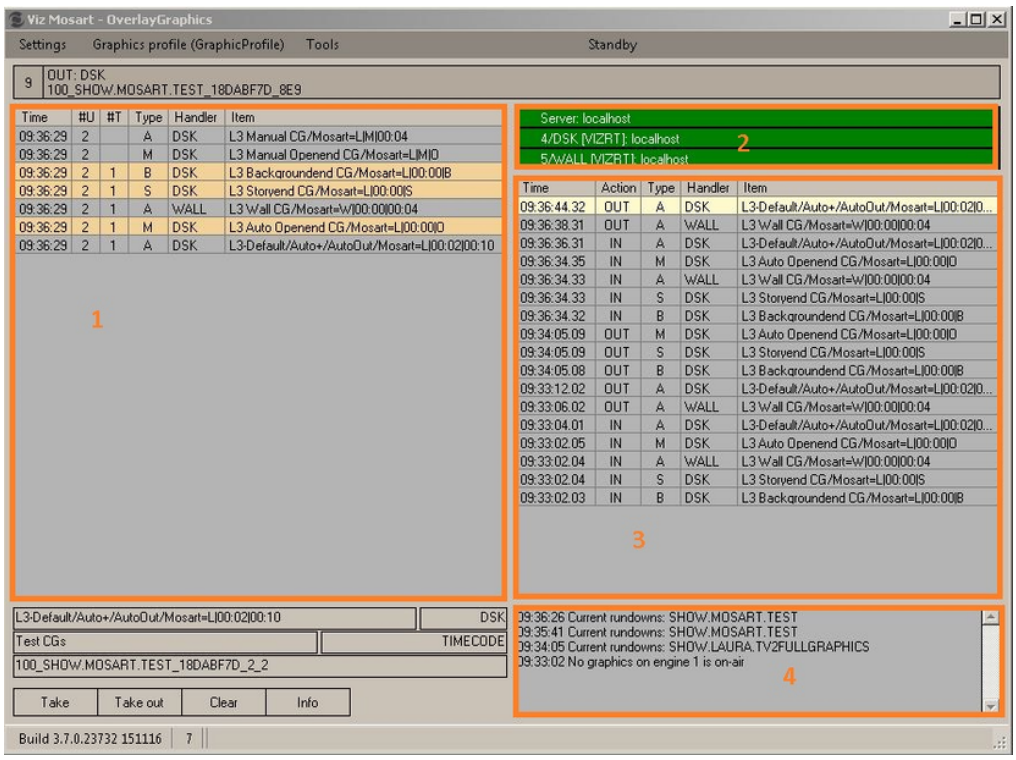
Overlay Graphics Interface

This is used for controlling and monitoring overlay graphics for graphics engines.

The two overlay graphics interfaces are:

- **OverlayGraphics** Interface (recommended).
- **Trio Interface** (deprecated, kept for backward compatibility only).

> **Info:** The Overlay Graphics Interface is only used for overlay graphics. Full-frame graphics are controlled through AV Automation.

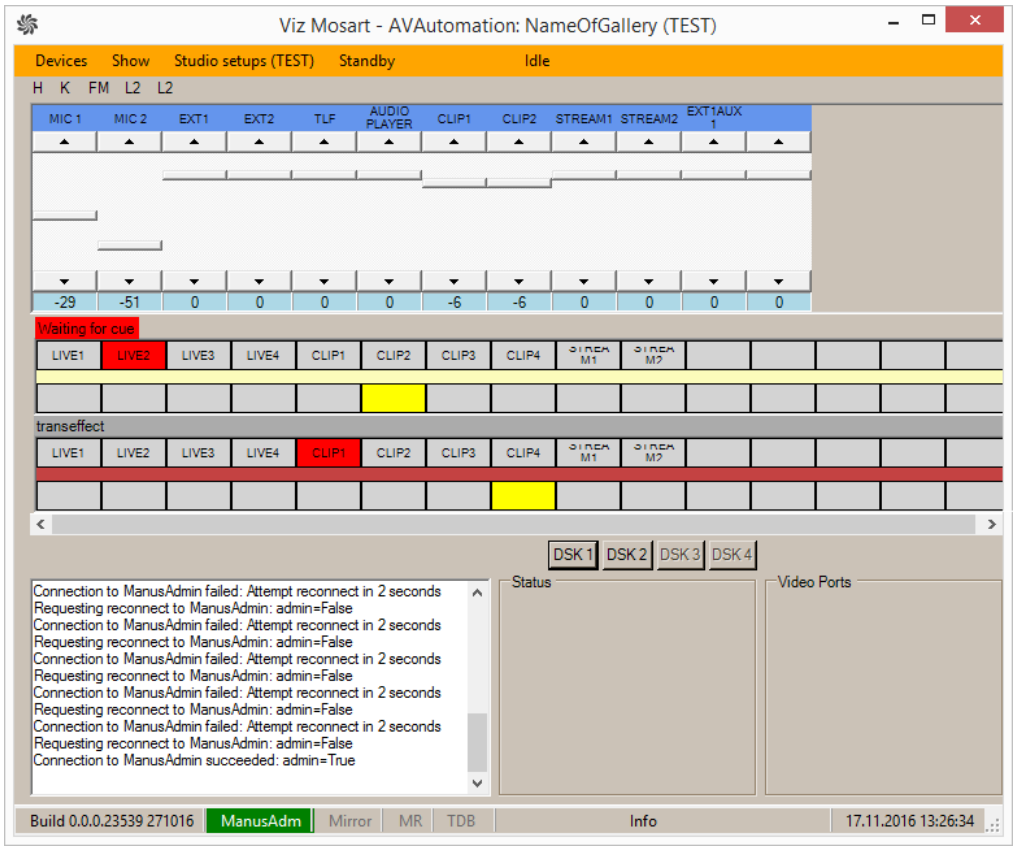


The Overlay Graphics Interface is used for controlling and monitoring overlay graphics for all Viz Mosart approved graphics engines. Configure it using Overlay Graphics Configuration. For a list of supported graphics devices, see Overlay Graphics Types.

Trio Interface

> **Info:** The **Trio Interface** is no longer recommended for use and is kept for backward compatibility only. Vizrt recommends using the Overlay Graphics Interface instead.

AV Automation



AV Automation (MMAVAutomation) controls all Viz Mosart broadcast devices with the exception of Overlay Graphics engines. Full-screen graphics are also controlled here.

Commands are issued to each device either on the fly through the Viz Mosart GUI, or as pre-determined by the rundown submitted from the NRCS.

All device commands are stored as predefined Viz Mosart templates in the **Template Editor** and are saved in `C:\channeltemplates` or in a template database.

Log Service and Log Viewer

Log Service

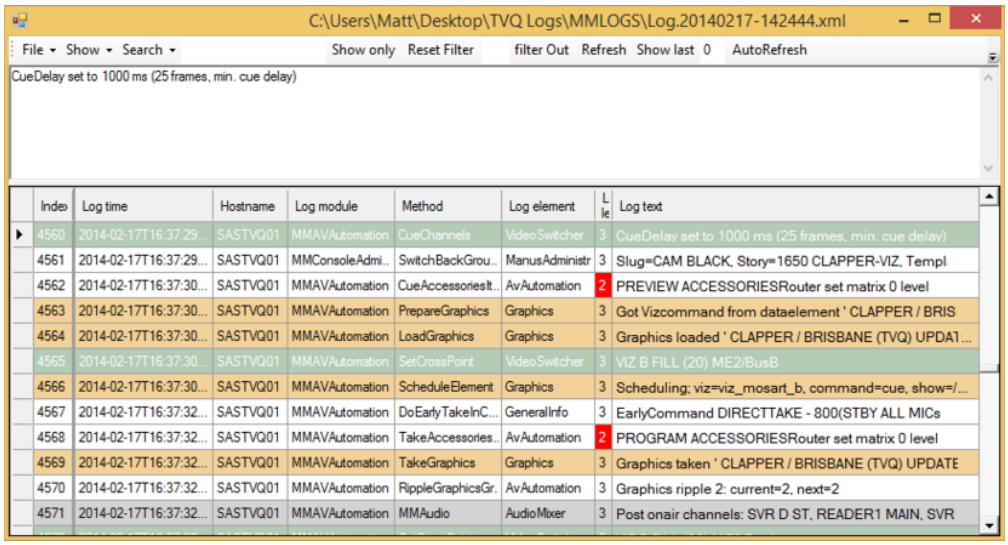
This is used for generating Viz Mosart command logs and is automatically installed together with the Viz Mosart Server and the Viz Mosart GUI.

Log Viewer

Logviewer

Log Viewer is used to decipher **Viz Mosart** logs. It is stored as an XML file once published, and as a .log file while actively running.

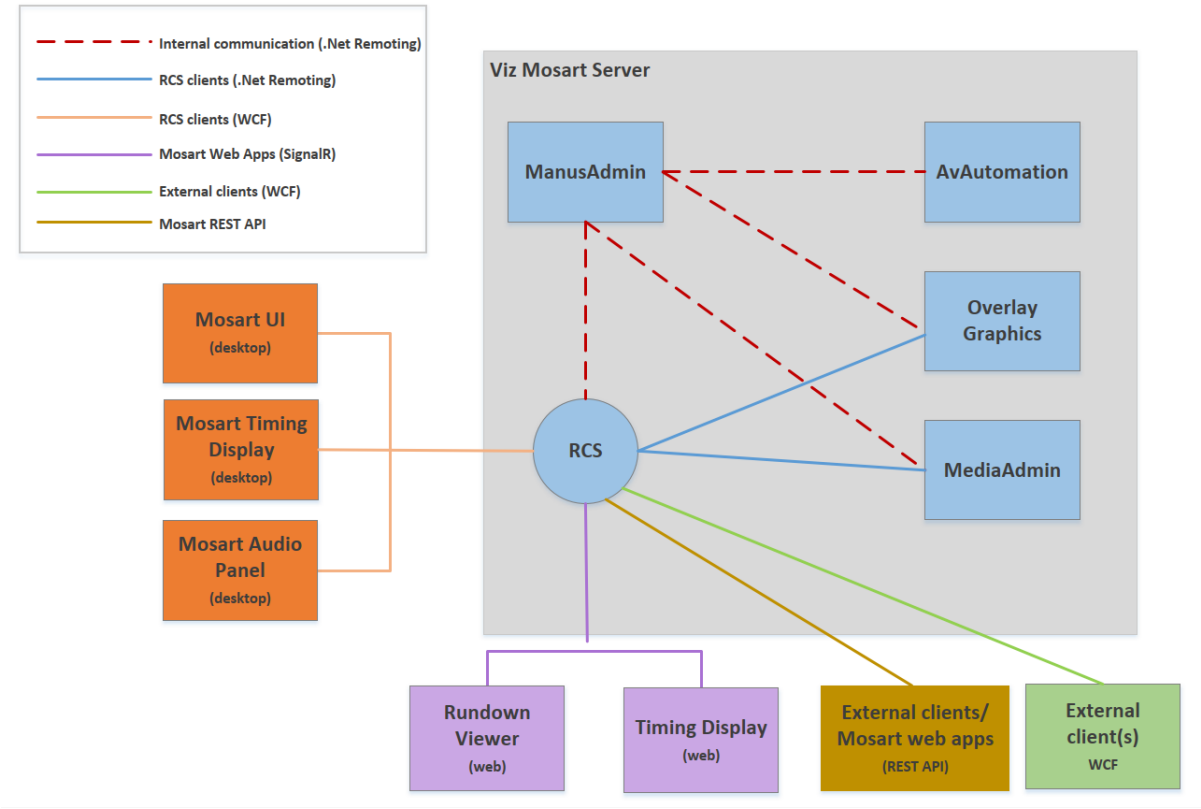
The log entries are color-coded to allow for quick command visualization in the file. Microsoft Excel may also be used to display log files (XML), and lets you use several layers of filtering and plot graphs.



Remote Control Service

The Remote Control Service (RCS) allows all external PCs, such as the Viz Mosart GUI and Timing Display, to connect to the Viz Mosart server. RCS is automatically installed together with the Viz Mosart server.

Copyright © 2026 Vizrt



Mosart Control Remote REST API

See more details here.

WCF

RCS provides a WCF (Windows Communication Foundation) interface for remote clients. This interface enables connection of external applications or services to the Viz Mosart server.

Configuration

For configuration options, see the comments in the associated standard configuration file *RemoteDispatcherServiceConfig.xml*.
This file is by default located at
C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles\RemoteDispatcherServiceConfig.xml

Security and connection stability

The settings file *RemoteDispatcherServiceConfig.xml* includes **SignalRAllowedOriginsin** for enhanced security and connection stability between the Mosart web server and the RCS. With this setting, you list valid origins that may access the SignalR service.

Any issues with the Mosart web server connections to the RCS, may be due to the RCS not automatically detecting the fully qualified domain name (FQDN) of the machine where it is installed.

- To resolve this, manually configure the SignalRAllowedOrigins in the RCS config file to include the fully qualified domain name of your Viz Mosart server.

Info:

- For Mosart applications like the Rundown Viewer and Timing Display (which rely on SignalR for real-time communication), when failure to recognize a FQDN prevents communication to RCS, defining **SignalRAllowedOriginsin** is essential.
- Multiple origins are specified either comma, semicolon or space separated.

For detailed configuration instructions, please refer to the RCS configuration file (*RemoteDispatcherServiceConfig.xml)* with commented documentation.

See Also

- Manus Administrator Configuration
- AV Automation
- AV Automation Device Properties
- Media Administrator Configuration

Copyright © 2026 Vizrt

2.4 Other Viz Mosart Applications

This section contains the following topics:

- Media Router and Mosart Template Database
- Omnibus Controller
- SNMP Service
- Test Suite

2.4.1 Media Router and Mosart Template Database

Media Router and Template Database with a MySQL database enables two features within Viz Mosart:

- Resource sharing: video servers, graphics devices, robotics, and so on.
  Viz Mosart can be configured to use shared broadcast devices between control rooms via the dedicated Media Router (MMR). This device enables Viz Mosart to select other devices before the next show.
  The vision switcher and audio mixer are not included in Media Router control, but may in some situations be shared, depending on the device's capability and/or the planned use of the device.
- Template sharing: Viz Mosart template information between multiple Viz Mosart installations.

These features are a licensed addition to the Viz Mosart Server. Please consult your nearest sales agent for more information.

2.4.2 Omnibus Controller

The Omnibus Controller allows communication with Omnibus systems for Viz Mosart installations running Omnibus equipment.

2.4.3 SNMP Service

This application allows for system monitoring of the Viz Mosart Server with SNMP agents such as Nagios and Navigator. Any SNMP agent application with MIB support is able to use this service.

2.4.4 Test Suite

The Test Suite contains various diagnostic utilities that can be used when installing Viz Mosart for the first time or when connected broadcast devices must be checked.

This is a list of some of the tools available in the Test Suite:

- **AsRunLogTester:** Test AsRun Log publishing
- **MediaRouterTester:** Test Media Router
- **MVCPSimul:** Test MVCP protocol
- **NexioTest:** Test Nexio protocol
- **QuantelTest:** Test Corba protocol
- **RoboTest:** Test camera robotics
- **ShotokuTester:** Test Shotoku camera robotics
- **TestAudioMixer:** Test Audio Mixer communications

Copyright © 2026 Vizrt

- **TestMIDIShowControl:** Test MIDI Lighting control
- **TestRouterControl:** Test Router communications
- **TestVDCP:** Test VDCP protocol
- **TestVisionMixer:** Test Vision Mixer communications
- **VintenTester:** Test Vinten 200 protocol

Copyright © 2026 Vizrt

2.5 Viz Mosart Clients

- Viz Mosart GUI
- NRCS Plugin
- Timing Display
- Rundown Viewer
- iNEWS Timer
- Audio Panel
- Audio Player

Info:

- Multiple instances of the above applications may run on any workstation connected to the same network as the Viz Mosart Server.
- All instances of Viz Mosart software must run the same Viz Mosart version with a standard OS across all Viz Mosart client machines.

2.5.1 Viz Mosart GUI

The Viz Mosart GUI workstation is a standalone networked PC with a single instance of the Viz Mosart GUI application (MosartMultiGui.exe) run in full-screen mode.
More than one PC can run the same software component, for example, you can add a backup GUI PC.

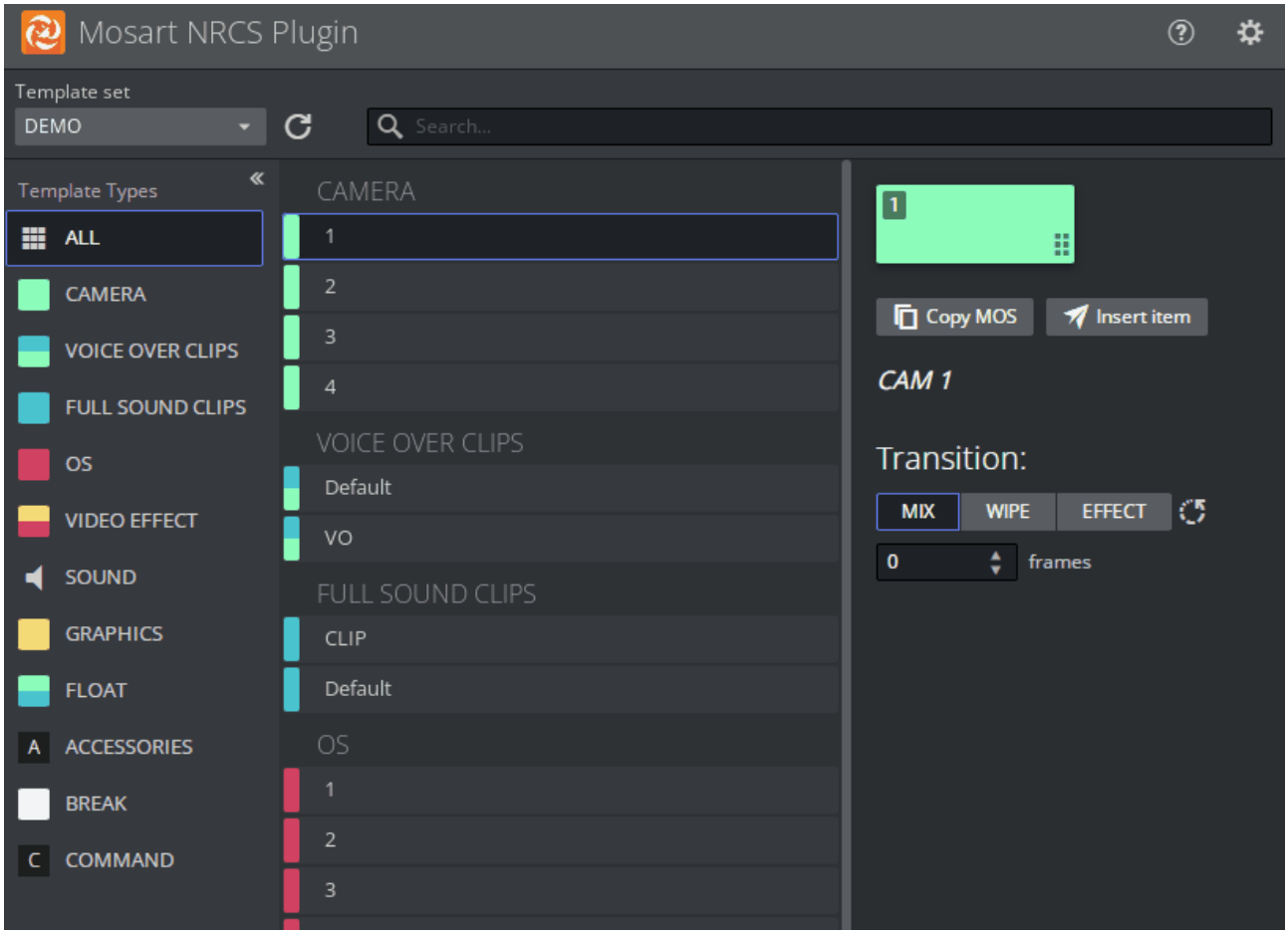
The Viz Mosart GUI is the main control interface for the Viz Mosart Server. The Director can run the production from a single key press on their workstation keyboard.

Alternate configurations may also host a Fader panel and a Timing Display. Each Viz Mosart Client application should run on a standalone networked PC.

The Viz Mosart GUI is described in detail in the *Viz Mosart User Guide*.

2.5.2 NRCS Plugin

The NRCS Plugin is one of the Mosart Web Apps. It delivers the functionality previously provided by the legacy Viz Mosart ActiveX Plugin.



Features include:

- Searching for a Template
- Modifying a Template
- Adding Template Details to the NRCS
- Template Change Alert

- Optimizing Your NRCS Plugin Workspace
- Copying the MOS Object Details
- Modifying Existing MOS Objects.

For details, please refer to the Mosart Web Applications Guide .

2.5.3 Timing Display

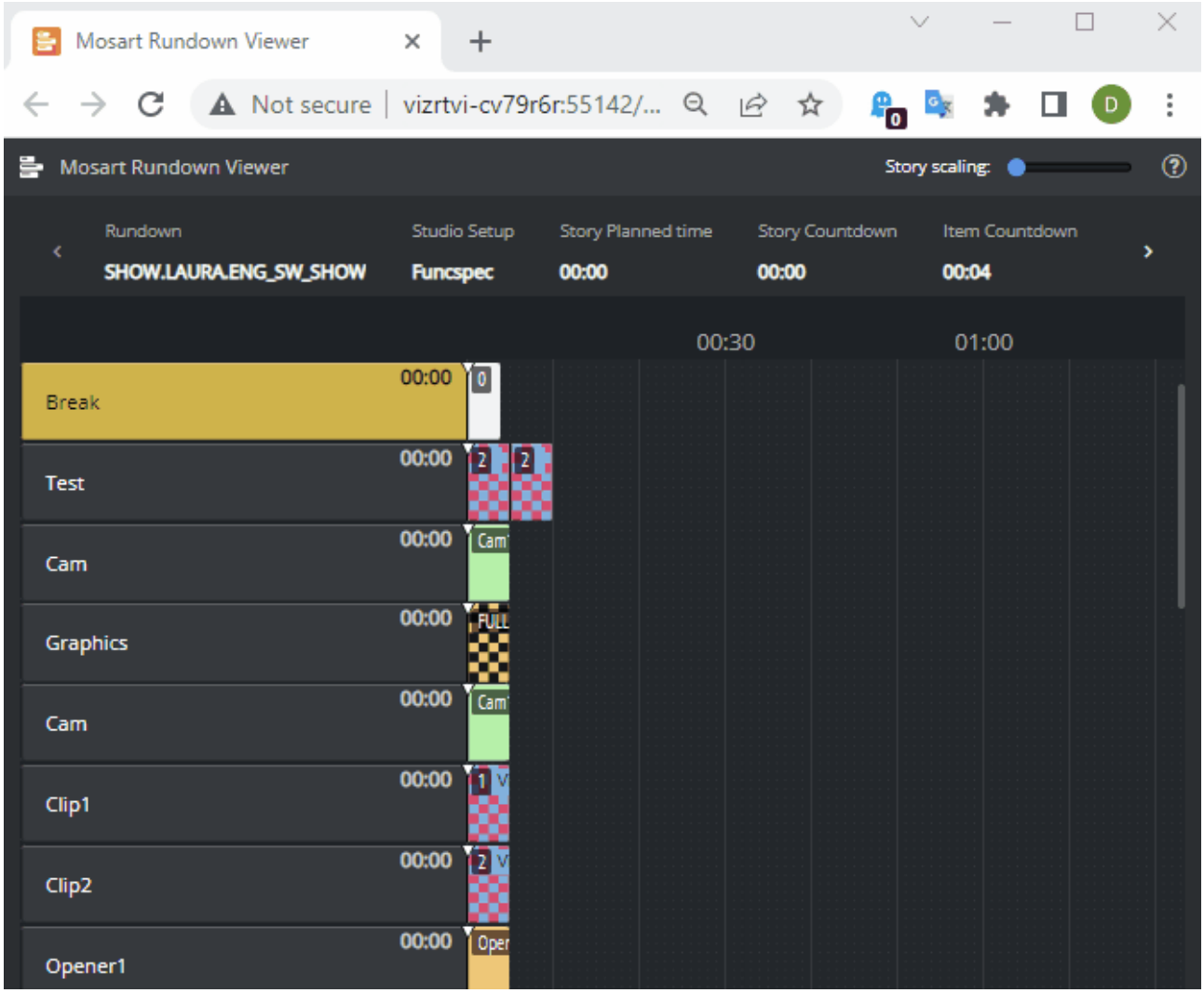
The Timing Display Web App replaces the legacy standalone utility Timing Display (WPFTimingInfo).

- The Timing Display provides timing information to the studio control room, studio floor or any device connected to the Viz Mosart network.
- Details are synchronized to the current rundown in the Viz Mosart GUI.
- Multiple Timing Displays can be customized to the specific needs of the production staff.

For details, please refer to the Mosart Web Applications Guide.

2.5.4 Rundown Viewer

The Mosart Rundown Viewer provides a convenient web browser view of details from the current Viz Mosart rundown.



- Displays the active Viz Mosart rundown.
- Summary of story timings.
- Supports a variety of devices. For example an iPad or networked phone.

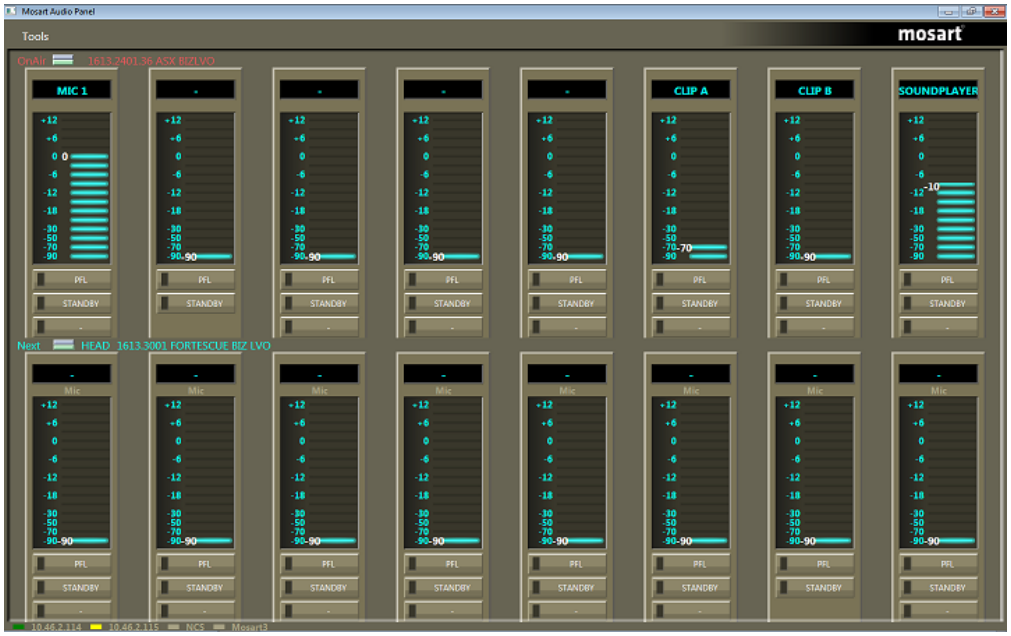
For details, please refer to the Mosart Web Applications Guide.

2.5.5 iNEWS Timer

The iNEWS Timer provides accurate timing of the production in iNEWS, and is used to synchronize the story currently on air in the Viz Mosart GUI, to the corresponding entry in iNEWS.

- See section iNEWS Timer.

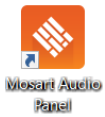
2.5.6 Audio Panel



Audio Panel (Server)

The Audio Panel (AudioPanel) lets you use a Behringer BCF2000 or JL Cooper MXL with a Viz Mosart Server. The panel controls data transmission between the physical hardware panel and AV Automation on the Viz Mosart Server.

Audio Panel (Client)



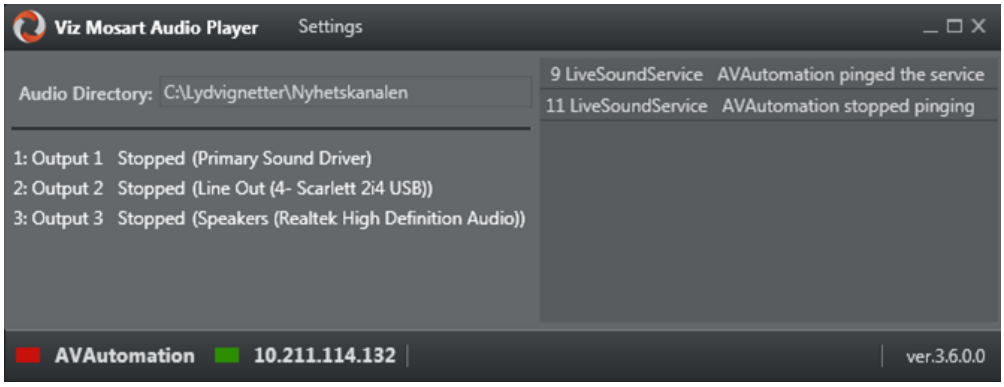
The Audio Panel (Client) is a software representation of the audio mixer connected to the Viz Mosart Server, displaying current faders, on air faders, and faders in preview.

The Audio Panel is described in detail in the *Viz Mosart User Guide*.

2.5.7 Audio Player

The Audio Player is part of the Viz Mosart installation and may be used to play out audio files located on the file system. This is useful for playing out files that are used on a regular basis, such as openers and audio-beds.

> **Info:** A broadcast sound card is required for audio output. If you're running the Audio Player on a Windows Server, you must install Windows Audio features on the machine in order for the various audio formats to work.



> **Info:** The Audio Player replaces the discontinued Soundfile Player.

- Viz Mosart can work with several commercial audio players.
- Operating the Audio Player is described in detail in the *Viz Mosart User Guide*.

3 Installation

This section contains an overview of how to install the various Viz Mosart Applications, by running standard Microsoft installers (.MSI files).

- Prerequisites
- Redundancy
- Licensing
- Viz Mosart Installation Files

3.1 Prerequisites

There are several system prerequisites that must be in place before the Viz Mosart Applications are installed.

This section contains:

- System Requirements
  - Microsoft .NET Framework
  - Microsoft Visual C++ Redistributable Package
- Hardware Requirements
  - Viz Mosart Automation Servers
  - Viz Mosart GUI PCs
  - Viz Mosart Audio Player
  - Viz Mosart Timing Display and Audio Panel PCs
  - Viz Mosart Shared Template Database
  - Pre-Installation Tasks
  - Connections
- Licensed and Optional Features

3.1.1 System Requirements

The following apply all machines running Viz Mosart software:

Prerequisite	Target	Description
Microsoft Windows Server 2016 or later	Viz Mosart Server	Operating system. Viz Mosart has been verified working with Windows Server versions up to and including Windows Server 2025.
Microsoft Windows 10 or later	Viz Mosart Client	Operating system.
Microsoft .NET Framework	All	See the *Viz Mosart Release Notes* for the applicable Microsoft .Net version. Most of the Viz Mosart installers cancel the installation process if the correct .NET version is not detected.
Microsoft Visual C++ Redistributable (both x86 and x64)	Viz Mosart Server	See the *Viz Mosart Release Notes* for the applicable Microsoft Visual C++ Redistributable Package. The Viz Mosart Installer will automatically add/update this software on your installer machine.

Copyright © 2026 Vizrt

Prerequisite	Target	Description
Microsoft Edge WebView2 Runtime (x64)	All	Required for the Mosart Central Settings which can be opened from the combined Manus Administrator (introduced in Viz Mosart 5.2.0) or AV Automation driver properties (since Viz Mosart 5.3.0) on the Viz Mosart Server. From Viz Mosart 5.4.0 also required for the Viz Mosart UI for doing Vizrt Graphics editing and for the web-based Timeline window and Story Script window introduced in that version.
Wibu-Systems CodeMeter Runtime Kit	Viz Mosart Server	Required for licensing
Windows Folder Structure	All	The following folder must be available, containing Viz Mosart installers, before installation: C:\mosart\installers
Windows Firewall	All	For some versions of Windows, it may be necessary to open ports. Either completely disable Windows firewall, or alternatively open for Ping and ports 8080-8099. If Mosart Template Database is used, port 3306 also must be opened.
Windows Audio	Viz Mosart Server	If MIDI devices are connected via direct conversion serial adapters, Windows Audio feature must be enabled on the server. **Info:** If using Windows Remote Desktop, audio must be set to play on the server, not the remote PC, as the Audio configuration will otherwise be disturbed.
Windows Server Type	Viz Mosart Server	It may be favorable to configure the Windows Server running the Viz Mosart Server software to **Application Server**, as this automatically pre-configures several features and also automatically sets a detailed .NET configuration.
Server Specifications	All	For more details, see below.

Info: All prerequisites must be completed before installing any Viz Mosart components, both server and client applications.

Microsoft .NET Framework

Microsoft .NET must be installed on the system before installing any Viz Mosart Applications. See the *Viz Mosart Release Notes* for the applicable Microsoft .NET version.

To determine which Microsoft .NET Frameworks are installed

- If you do not know how to determine this, contact your IT System Administrator to learn how to.

Installing the Microsoft .NET Framework

1. Obtain the applicable Microsoft .NET Framework Installer from the Microsoft Download Center.
2. Run the installer and follow the prompts.
3. Restart the machine if prompted on completion.

Microsoft Visual C++ Redistributable Package

A *Microsoft Visual C++ Redistributable Package* must be installed on the system before installing the Viz Mosart Server software. See the *Viz Mosart Release Notes* for details on the Microsoft Visual C++ Redistributable Package.

> **Info:** Viz Mosart is a 32-bit program, but it also includes x64 components.

Obtain the applicable Visual C++ Redistributable Package (both **x86** and **x64**) installers from the Microsoft Download Center.

Installing Microsoft Visual C++ Redistributable Package:

1. When downloading, you have to download both the **32-bit (x86)** version and the **64-bit (x64)** version.
2. Run the installers and follow the prompts.
3. Restart the machine if prompted on completion.

3.1.2 Hardware Requirements

Viz Mosart Automation Servers

- When operating with (optional) a *backup* server alongside *main*, each server needs its own machine/instance.

Minimum Recommended Hardware Configuration

- Dual Power Supply for redundancy
- Redundant disk configuration
- Integrated Lights Out (iLO), IPMI or similar with Keyboard-Video-Mouse (KVM) support
- Windows Server operating system with applicable .NET Framework
- 2 x Gigabit Ethernet ports
- 1280 x 1024 display adapter
- 16 GB Memory

Recommended Performance-related Components

- 2 Processors with performance as 2.6 GHz 6-core and with 12 MB L2 cache
- Minimum 50 GB free storage space

Copyright © 2026 Vizrt

- Windows Server 2016, 2019, 2022 or 2025 operating systems

Site-specific Requirements

Location-specified external device connections might include the following examples:

- PCI multiport serial card RS-422/RS-232 (short distance connections).
  www.moxa.com CP-118EL-A.htm
- IP connected Terminal Servers with serial output ports RS-422/RS-232 (long distance connections).
  www.moxa.com - 2650
  www.moxa.com - 5150
- IP connected Terminal server with MIDI output port.
  https://kiss-box.com/product/midi2tr-dual-midirtp-midi-transceiver/
- Additional Ethernet ports
- IP connected GPIO control box
  https://www.wut.de/e-57730-ww-daus-000.php

Info: W&T is the only GPIO control device vendor supported by Mosart.

- Up to four boxes can be controlled, giving 48 outputs.
- Only the first box is used for input, giving you 4 predetermined Mosart rundown functions, and 8 general inputs.
- Display, keyboard and mouse should be available in the relevant gallery.

Viz Mosart GUI PCs

Whilst not mandatory, it is recommended to have both a main and backup client. Main and backup require their own individual Windows machine.

Minimum Recommended Hardware Configuration

- Local PCs physically located in the control room. Alternatively Integrated Lights Out (iLO), IPMI or similar, optionally with Keyboard-Video-Mouse (KVM) support.
- Windows 10 or Windows 11 64 bit operating system with applicable .NET Framework
- Gigabit Ethernet port
- Approved video adapter with resolution 1920 x 1080
- 32 GB Memory
- Displays with minimum 1920 x 1080 resolution, keyboard and mouse should be installed in the gallery.

Recommended Performance-related Components

- Processor with performance equal or better than Intel i7 2.6 GHz 6-core, 12 MB L2 cache
- Minimum 50 GB free storage space

Viz Mosart Audio Player

If you will use the Viz Mosart Audio player, a single dedicated PC is normally placed in the server room.

The degree of hardware redundancy required for this function shall be considered.

Copyright © 2026 Vizrt

The PC running the Viz Mosart Audio Player should be equipped with:

- A broadcast quality audio card registering with Windows driver.
  For example, Lynx Studio Technology AES16e PCI Express AES/EBU Interfaces, www.lynxstudio.com.
- Windows 10 or Windows 11 operating system with applicable .NET Framework.
- Ensure the audio card is supported by the selected Windows operating system.

Remaining hardware configuration and performance parameters are similar to GUI PCs

Info: The Viz Mosart Audio Player replaces the discontinued *Mosart Sound File Player*.

Viz Mosart Timing Display and Audio Panel PCs

The main purpose of these two PCs is to display the output from the respective Viz Mosart applications.

- For the Audio Panel it is also possible to interact with the application via mouse and keyboard.

The configuration and performance parameters shall be similar to the GUI PCs.

Viz Mosart Shared Template Database

The Template Database is built on an Oracle MySQL Server. Actual database traffic will be light – any modern PC will suffice, since Viz Mosart template features are more similar to a shared storage function than a classic database transaction server.

Recommended MySQL Software

- Refer to the *Viz Mosart Release Notes*.

Minimum Recommended Hardware Configuration

- Keyboard-Video-Mouse (KVM) support.
- Windows operating system with applicable .NET Framework.
- Gigabit Ethernet port.
- 1280 x 1024 display adapter.

The above hardware can be increased, for example with dual redundant PSUs, NICs, disks etc. for redundancy.

Info: Some of the hardware components (for example dual PSU) might necessitate an upgrade to a server proper, running a recent version of Windows Server.
It is also possible to add another machine as a backup server.

Recommended Performance-related Components

- Processor with performance equal or better than Intel 2.0 GHz Dual-core.
- 8 GB Memory.
- Minimum 50 GB free storage space.

Copyright © 2026 Vizrt

Pre-Installation Tasks

A computer platform as described must be installed by the customer prior to the start of installation.

This includes:

- Dual independent power feeds to Viz Mosart Servers and the GUI PCs, where one of the inputs is from an Uninterruptible Power Supply (UPS) source.
- Set-up of keyboard, video and mouse for Viz Mosart GUIs and Viz Mosart Servers in the relevant gallery.
- Cabling and connectivity-testing from the Viz Mosart servers to the gallery equipment to be automated.
- Tested networks connection from the Viz Mosart servers to the Newsroom system and the Viz Mosart GUIs.
- For all IP connections (see Connections below), the customer must make sure relevant ports and services are opened in any in-between firewalls.
- Viz Mosart Support should be permitted remote access to all Viz Mosart computers for online support.

Connections

All Viz Mosart applications and services are connected via TCP/IP. This allows them to run on any computer in a common logical network.

- Connections to external devices are a combination of TCP/IP and serial communication protocols like RS-232, RS-422 and MIDI.
- Extenders and translators, terminal servers and MIDI over IP, may also be used.

Info: All device connections (such as link layer and application protocols) are dependent on the device itself.

3.1.3 Licensed and Optional Features

For general Licensing guidelines, see section Licensing.

- **Engine Switcher**: If Viz Mosart is licensed to use this optional feature, please refer to the Viz Engine requirements in the *Viz Engine Administrator Guide* (version 5+).
- **Story Recorder**: Requires a separate license and specific setup, as described in section Story Recorder Setup.

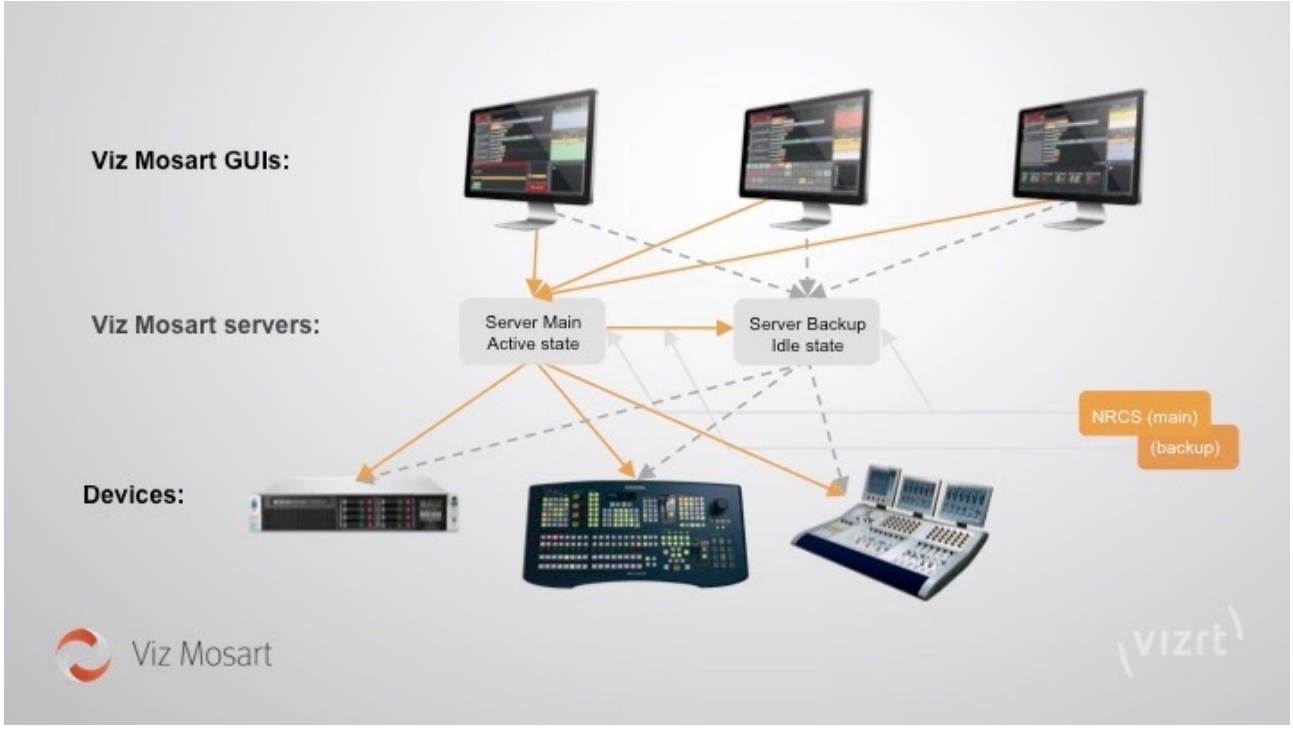
Copyright © 2026 Vizrt

3.2 Redundancy

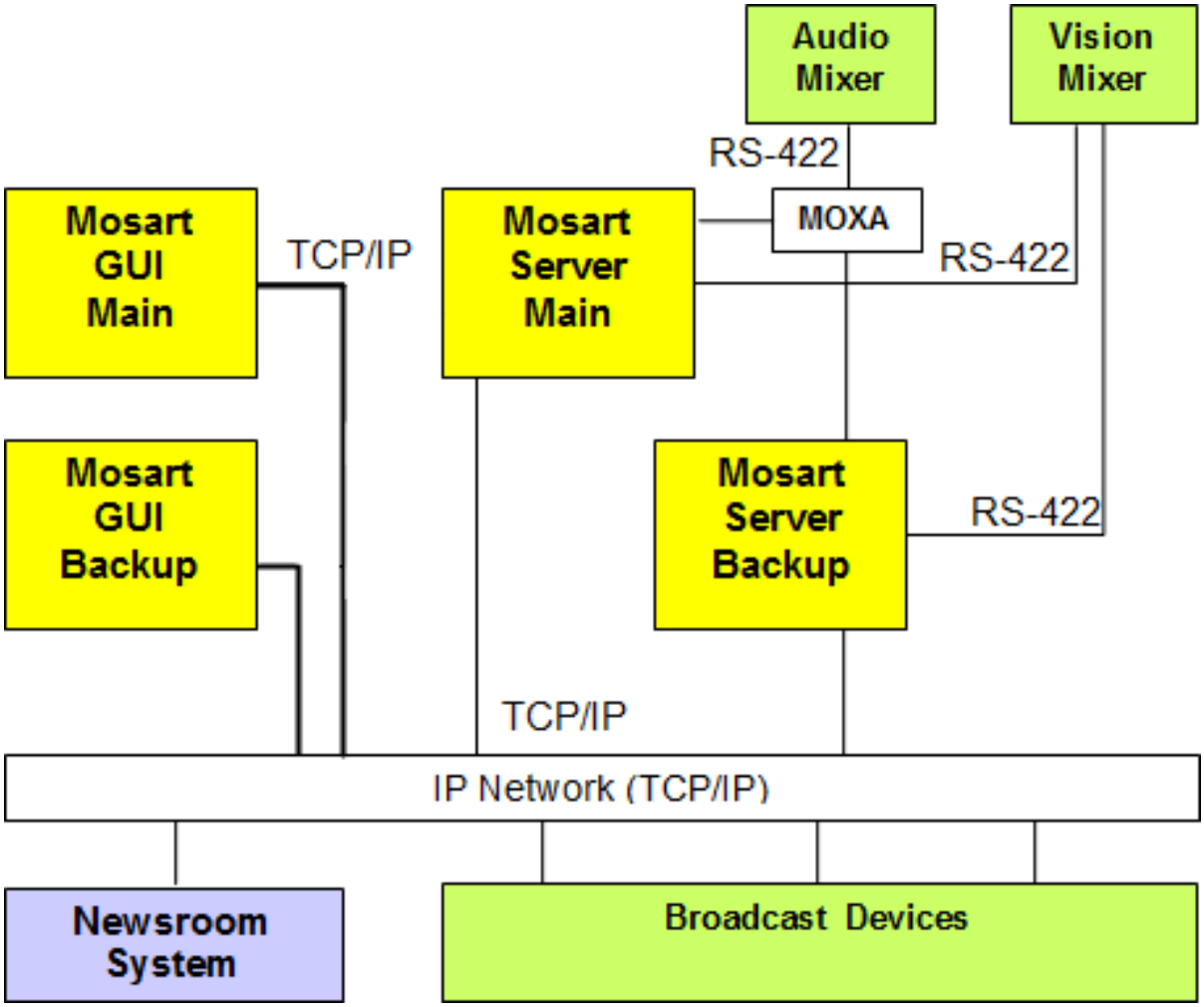
As all Viz Mosart applications and services are interconnected using TCP/IP, it is possible to run all components on different computers, allowing multiple redundancy setup possibilities.

- Automation Server Mirroring
- Connected GUIs
- Graphical Interfaces
  - AV Automation - The Engineering Interface
  - Viz Mosart GUI (Operator's Interface)
- Device-specific Redundancy
  - Video server redundancy
  - Video Switcher Backup
  - Newsroom System Switch-over

Viz Mosart is based on a dual redundant server and a GUI PC configuration with dual redundant connections (Network Teaming) to broadcast devices. The recommended redundancy setup is to run a Main and Backup Viz Mosart Server with two workstations running the Viz Mosart GUI.



An example of a recommended redundancy setup is shown schematically below (the actual solution will depend on the characteristics of the connected devices):



The system in the example above has the following characteristics:

- Dual redundant system.
- All Viz Mosart applications and services in the Viz Mosart Server suite are interconnected and running on the same server.
- Both main and backup Viz Mosart GUI applications are configured to both main and backup Viz Mosart servers. Initially they use the *main* Viz Mosart Server.
  Only one click is needed to instead connect them to the backup server.
- All external equipment and systems are shared between the main and backup servers.
- The vision mixer is connected using serial RS-422, with both main and standby cabling (as the mixer supports multiple serial connections).
- The audio mixer in this example has only one serial connection. A serial/ethernet converter (for example MOXA Terminal Server) has been used for enabling both servers to communicate with the mixer.
- All other equipment is connected via TCP/IP on a common network. IP redundancy is presumed on a network level through the use of Network Teaming.

Copyright © 2026 Vizrt

3.2.1 Automation Server Mirroring

The automation servers work in an active/idle setup, where the active server pushes the various states to the idle server. If the active server fails, the GUI operator can manually select the idle server, which then seamlessly takes over, becoming the active server.
In the case of a server failure, the servers operate independently of each other.

3.2.2 Connected GUIs

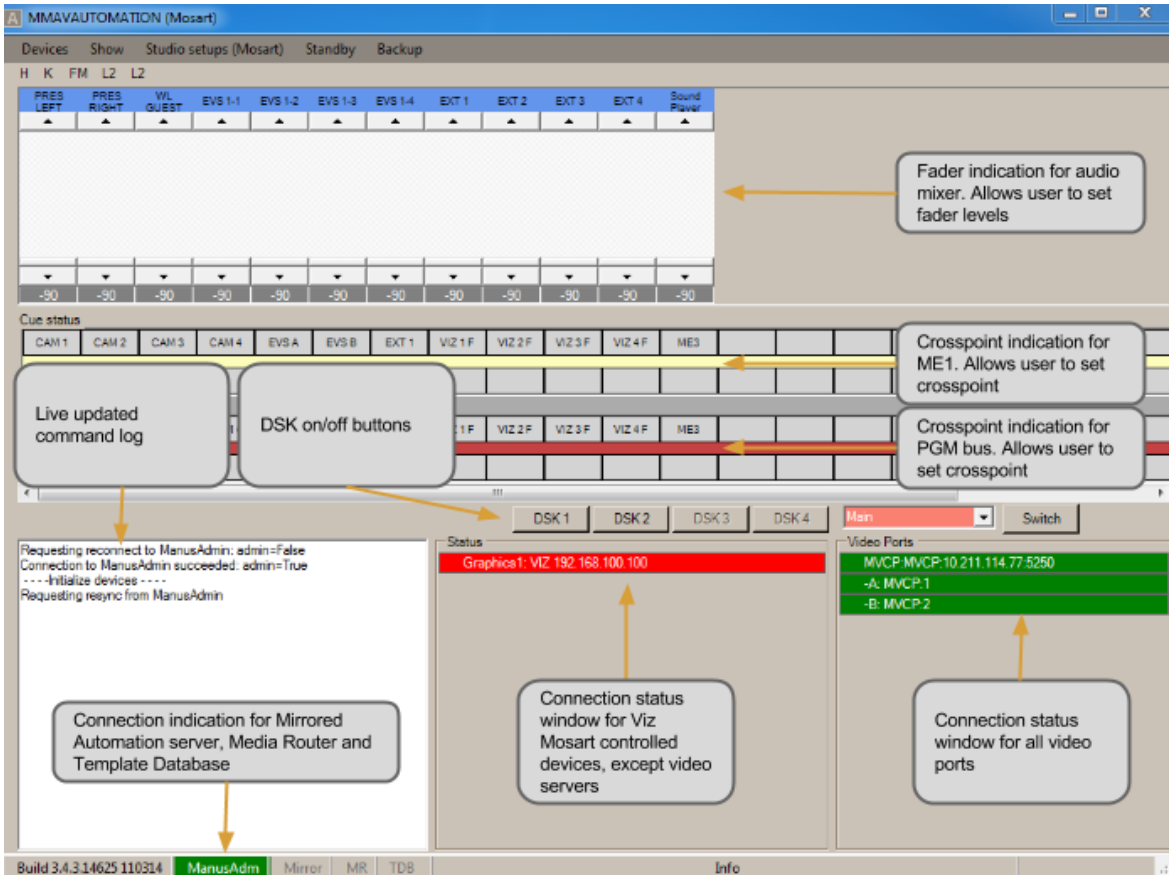
Viz Mosart supports an unlimited number of user interface clients. In the case of a server switch-over, all GUI clients follow this switch-over.

3.2.3 Graphical Interfaces

Viz Mosart has two main interfaces that expose redundancy features of the system to operators:

AV Automation - The Engineering Interface

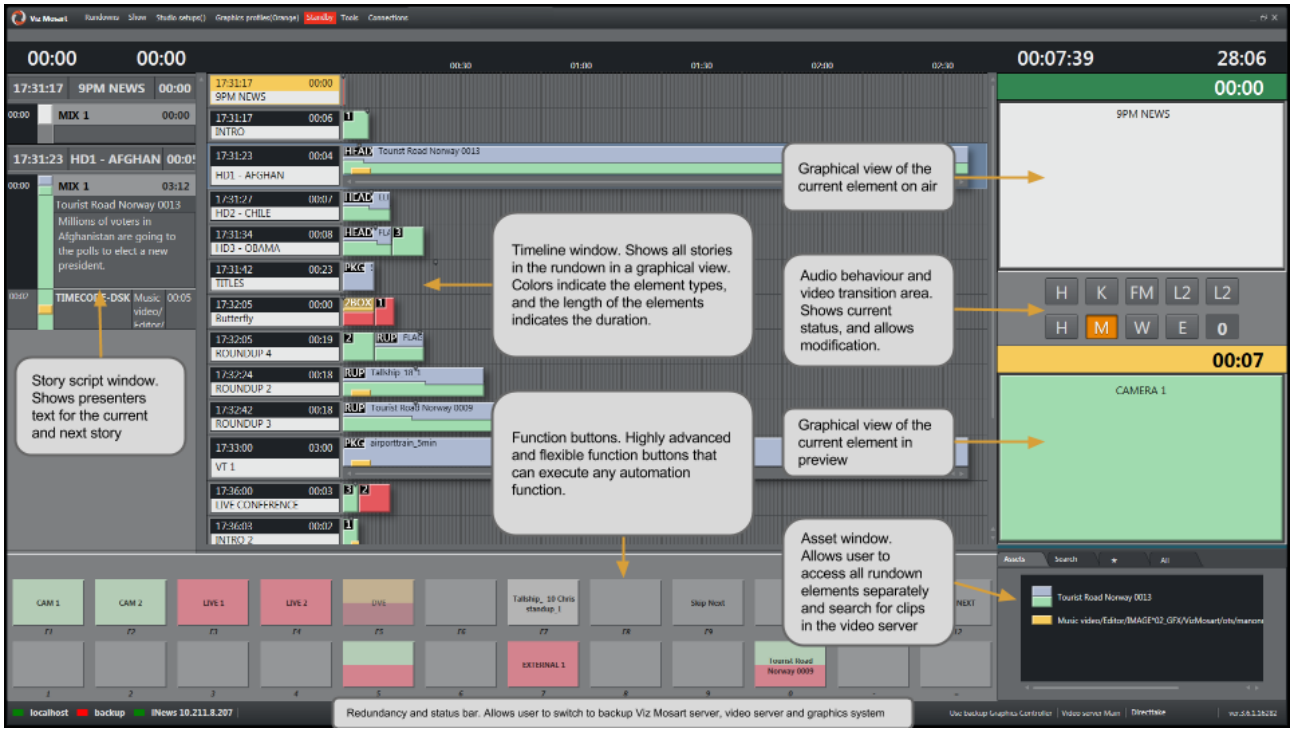
AV Automation provides connection statuses of the connected devices and enables the user to access several functions connected to system redundancy. AV Automation runs on each of the two automation servers.



Viz Mosart GUI (Operator's Interface)

While the GUI primarily is for on air operation, the *lower part* of the interface allows the operator to do the following:

- Switch between the automation servers (while the system is on air)
- Set video server salvos (for example main to backup salvo)
- Switch to the mirrored video servers seamlessly
- Switch to backup graphics controller.

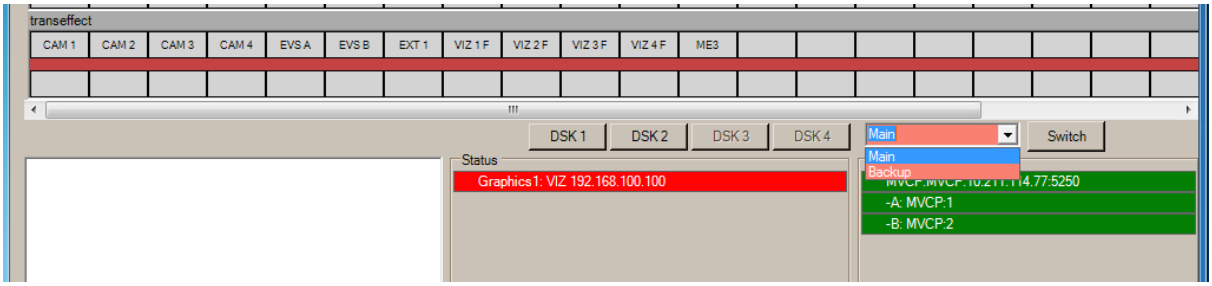


3.2.4 Device-specific Redundancy

Video server redundancy

Viz Mosart supports connection to unlimited amounts of video servers, configurable with the following possibilities:

- Multiple A/B ripple groups
- Mirrored video playout in any group
- Salvo setups for quick recalls of various video server control setups.
  Video server salvo switch-over is illustrated below.



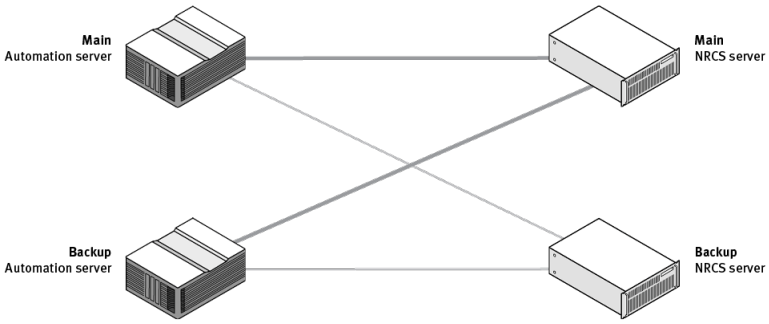
Video Switcher Backup

Viz Mosart supports router control which can be used as an emergency video switcher. The system allows the broadcaster to set crosspoints on the video router simultaneously with the video switcher. This can be used as an emergency output at video switcher breakdown.

Newsroom System Switch-over

Viz Mosart supports dual NRCS servers and will follow the backup switch-over mechanism of the newsroom system.

Viz Mosart



3.3 Licensing

In version 5.2.0 a license management backend, a License Configuration menu and associated Notifications were introduced.

In later versions, license-based feature control will be introduced. To ensure service continuity with the correct licenses in place before upgrading, please contact your Vizrt account manager.

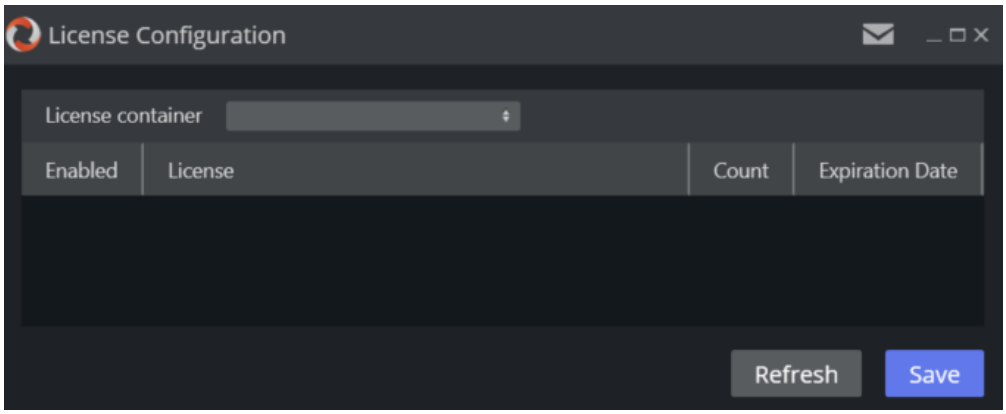
3.3.1 License Management

Vis Mosart uses the same Wibu-based backend licensing architecture as other Vizrt products.

- For further details, please refer to the latest version of the *Vizrt Licensing Administrator Guide*.
- An orientation course to the Management portal is also available on *Viz University*.

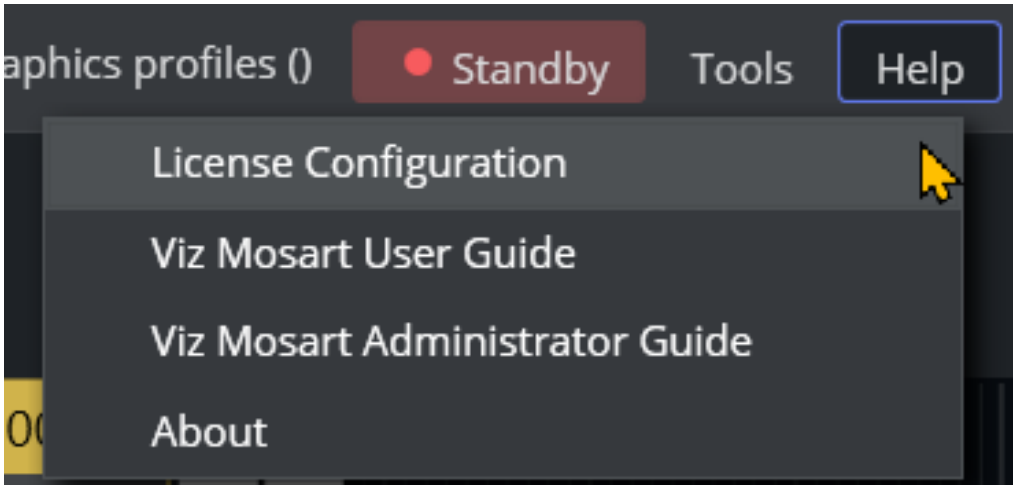
3.3.2 License Configuration status

The configuration status of Viz Mosart licenses can be seen in Viz Mosart UI, using the new License Configuration UI.



In version 5.2.0 this License Configuration UI is usually empty, and is present only for testing, and for administrators to become acquainted with the concept. In later versions it will display the active features that require a license, and the status of the respective licenses.

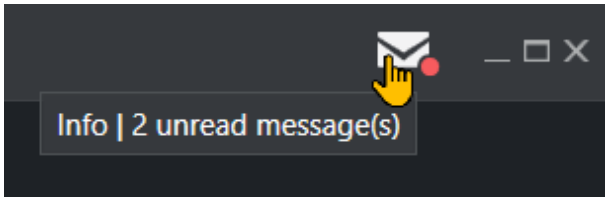
- To open the License Configuration UI from the Viz Mosart UI menu bar, navigate to **Help > License Configuration**.



3.3.3 License Notifications

Important licensing notifications are presented to the user through a the Viz Mosart UI.

A notifications icon appears at the top of all menus. A red marker indicates an unread notification.



- Clicking the icon opens the Notification Box menu, listing important system messages that will usually require action.

3.4 Viz Mosart Installation Files

Viz Mosart is delivered as a software package comprising various server and client applications, used for connecting to studio devices and managing the NRCS rundown, imported into Viz Mosart.

- Viz Mosart Installer Files
- Additional Mosart Files
- Installation and Upgrade
- Version Numbering

3.4.1 Viz Mosart Installer Files

Obtain the latest version of Viz Mosart at Vizrt's customer FTP:

1. Authenticate to Vizrt's FTP at https://download.vizrt.com/
2. Navigate to products/VizMosart/Latest Version/

The principle files (Mosart Server + GUI) are in *.exe* format, supporting utilities are of type *.msi*.

Installer	Machine Role	Description
VizMosartAudioPanel	Audio Client (GUI or standalone PC)	Installer containing the **Audio Panel**, an application for controlling audio faders, both on air and in preview.
VizMosartAudioPlayer	Viz Mosart Server or Audio Client	Installer containing the **Audio Player**, typically installed on a computer, or Viz Mosart Server, attached to the newsroom audio mixer. **Info:** Audio Player is a replacement for the discontinued Soundfile Player.
VizMosartGUI	Viz Mosart GUI Client	Installer containing the **Viz Mosart UI** application, which is the Viz Mosart user interface.
VizMosartINEWSTimer	iNews Client	Installer containing the **iNEWS Timer** (iNEWS Auto Timer), an application for creating accurate timing of the production in iNEWS. See section iNEWS Timer.
VizMosartLogViewer	Viz Mosart Server	Installer containing the **Log Viewer**, an application used to simplify reading of Viz Mosart logs.
VizMosartMediaRouter	Viz Mosart Database Server	Installer containing the **Media Router**. For more details, see *Media Router* and *Mosart Template Database Administrator Guide*.

Installer	Machine Role	Description
VizMosartOmnibusController	Viz Mosart Server	Installer containing the **Omnibus Controller**, an application used for Omnibus Server control.
VizMosartServer	Viz Mosart Server	Installer containing the various Viz Mosart Server components: Manus Administrator, Media Administrator, AV Automation, and Overlay Graphics Interface.
VizMosartSNMPService	Viz Mosart Server	Installer containing the **SNMP Service**, used for monitoring the Viz Mosart Server suite via SNMP.
VizMosartTestSuite	Viz Mosart Server	Installer containing the **Test Suite**, test utilities used when installing Viz Mosart.
VizMosartTimingDisplay	Timing Client	Installer containing the **Timing Display**, used to provide timing information to the studio control room and studio floor. Is superseded with the *Web Application* **Timing Display**. See Web Applications

3.4.2 Additional Mosart Files

Obtain the latest version of additional Viz Mosart software at Vizrt's customer FTP:

1. Authenticate to Vizrt's FTP at https://download.vizrt.com/
2. Navigate to products/VizMosart/Latest Version/

Installer Location	Machine Role	Description
WebApplications	Viz Mosart Server	Installer containing a Configuration tool and all Mosart Web Applications. Mosart Web Applications are described here.
WebApplications/StreamDeckPlugin	Viz Mosart Server	Directory containing Elgato Stream Deck plugin for Viz Mosart. Details here.
../Engine Switcher Scenes	Viz Engine Server	Directory containing sample scenes for Engine Switcher. Located at products/VizMosart/Engine Switcher Scenes. See also section Viz Engine as a Switcher.

3.4.3 Installation and Upgrade

Installing or upgrading Viz Mosart preserves any stored settings and configurations.

You run the Viz Mosart Installation Files manually from Windows Explorer.

Installing for *Current user only* or for *Anyone who uses this computer*

Normally, Viz Mosart applications are installed for *Anyone who uses this computer*, but in some situations it is preferable to install for *Current user only*. If you have have a previous installation for *Anyone who uses this computer* and then install for *Current user only*, Windows' *Apps & Features* will show *two* installations of the application, both the previous one and the new one, and you will also see two shortcuts on your Desktop. This is a feature of Windows, but can lead to confusion with subsequent updates.

- A recommendation to avoid this particular situation is to *uninstall* the previous Viz Mosart application before installing the new one.

Warning: When installing to *Current user only*, the executables are, by design[*], installed in the folder "%ProgramFiles(x86)%\Mosart Medialab", and *not* in the location dedicated to the Current user (normally *%localappdata%\Programs*).

- This means that if a user no longer needs to start Viz Mosart applications, do not uninstall the applications for that user only, since you will then effectively uninstall it for **all** users! Instead, simply remove the application shortcuts for that user.

[*] The various Viz Mosart applications shall normally always be on the same version, to ensure compatibility of features. If the Viz Mosart applications were installed under the various users' dedicated locations, then during an upgrade, you would have to repeat the installation of a new version for all Viz Mosart users for that computer. In addition to the time and effort this would require, you risk that some users might miss the upgrade. When one of these users later starts a Viz Mosart application, for example the Viz Mosart GUI, it would no longer be compatible with the commonly upgraded Viz Mosart applications, like the Viz Mosart Server.

Backing-up Viz Mosart

The locations are:

All files in the following folders:

- C:\channeltemplates
- %localappdata%\Mosart_Medialab
- %programdata%\Mosart Medialab\ConfigurationFiles
- %programfiles(x86)%\Mosart Medialab\<Mosart application>\ConfigurationFiles
- All files with extension .exe.config in folders
  %programfiles(x86)%\Mosart Medialab\<Mosart application>\
  where <Mosart application> is the relevant Viz Mosart application, for example, Mosart Server, Mosart GUI.

Windows registry settings for

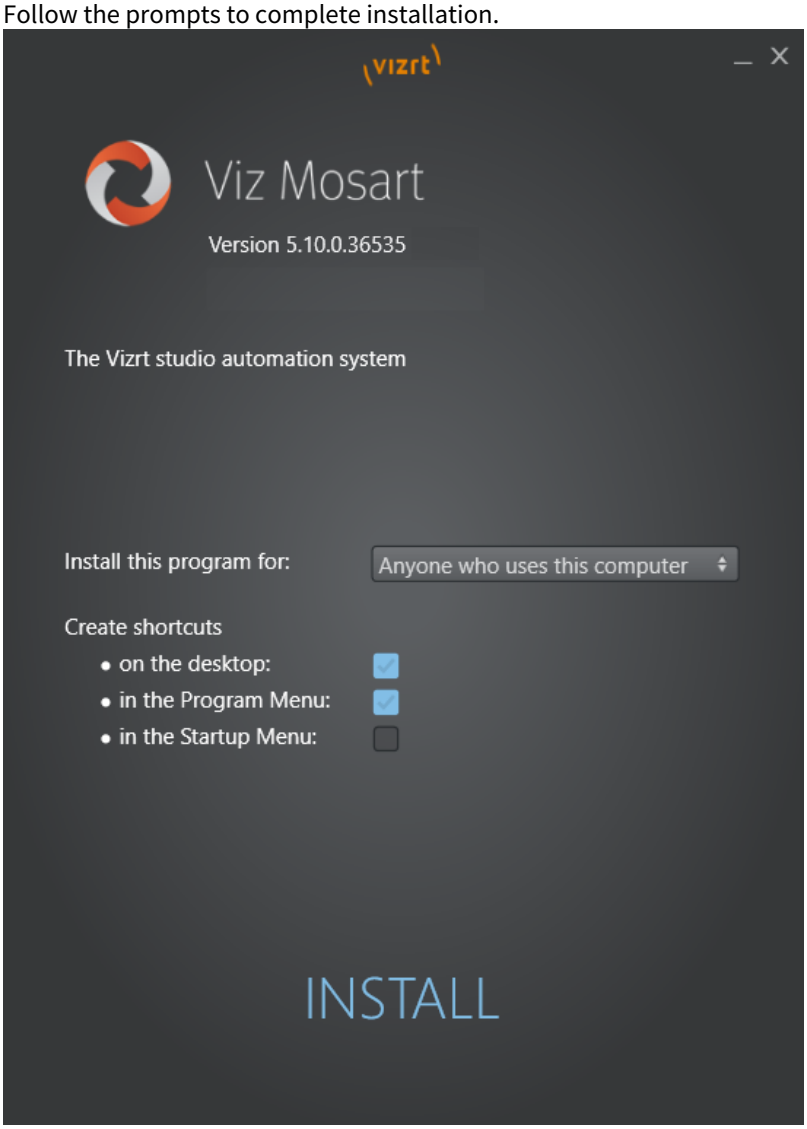
- HKEY_CURRENT_USER\Software\[Wow6432Node\]Mosart Medialab
- HKEY_LOCAL_MACHINE\Software\[Wow6432Node\]Mosart Medialab

Installation

To install Viz Mosart

> **Info**: Only the combined Viz Mosart Manus Administrator is installed. The selection previously made with the **NRCS Type** drop-down is no longer required.

1. Download all relevant Viz Mosart Installation Files to your preferred location.
2. If you are upgrading from an earlier Viz Mosart version it is recommended to take a backup of your current installation settings before you start the installation of the new version. See Backing-up Viz Mosart above.
3. Double-click the installation files (msi or exe).
4. Follow the prompts to complete installation.



3.4.4 Version Numbering

Installation folders are named with the Viz Mosart version number. All files inside that folder also have the corresponding version number attached to the filename.

Version numbers have the following syntax:

- Major.Minor.Revision.Build (for example **5.12.5.12345**)
  Where:
  Major.Minor.Revision denotes the Viz Mosart Version Number.
  Build denotes a build number.

Example: `Installers\5.12.5.12345\VizMosartServer-5.12.5.12345.exe`

In this example, the download comprises a folder for the Viz Mosart 5.12.5.12345 release, containing associated installers.
Every significant code change of Viz Mosart results in a change to the Viz Mosart version number. This increases either the major, minor, or revision number, depending on the impact of the code change.

Copyright © 2026 Vizrt

4 Security in Viz Mosart Ecosystem

- Introduction
- What is secured, and by what
- How the layers fit together
- Roles across Viz Mosart
- Setup at a glance
- End to end verification
- Troubleshooting

4.1 Introduction

Parts of Viz Mosart can be run with encrypted transport (HTTPS) and user sign-in (OpenID Connect). This chapter describes how the security layers of a Mosart installation fit together, how to choose an authentication mode, and how to set everything up: the identity provider, the certificates, the Mosart Web Apps and the Mosart servers.

> **Note**: Before taking your system into production, review the default security settings. Out of the box, neither HTTPS nor authentication is enabled, and the Mosart REST API accepts any caller that presents the shared REST API key.

The chapter has four sections:

- Choosing an Authentication Mode: default, API key or OIDC, and what each mode means for your integrations.
- Identity Provider and Certificates: what your IT department prepares and hands back.
- Securing the Mosart Server: HTTPS and OIDC on the main and backup Mosart servers.
- Password-Protected Features: the individually protected features that exist independently of HTTPS and OIDC.

4.2 What is secured, and by what

HTTPS and OIDC authentication secure the surfaces reachable from a browser or from another system: the Mosart Remote Controller Service, which serves the Mosart REST API and the SignalR hub, and the Mosart Web Apps with their proxy. These are the ways into a Mosart installation from outside the automation network, which is why authentication is implemented here first.

The rest of an installation is protected by other means:

- **The Viz Mosart desktop applications**, meaning AV Automation, Manus Administrator, Media Administrator and the Mosart GUI, run inside the automation network and are protected by that network's isolation, together with the password-protected features described in this chapter. They do not take part in OIDC sign-in.
- **Links between Mosart components**, such as Manus Administrator talking to the Remote Controller Service, use their own internal mechanisms rather than the OIDC path.
- **Device and newsroom protocols**, meaning MOS, GPI/O, and serial or TCP device control, follow the specification of each protocol and device. Where a device offers a secured connection, Viz Mosart can be

configured to use it: the Generic REST driver, for example, supports HTTPS with a choice of authentication types.

> **Note**: Plan the automation network as a trusted zone. HTTPS and OIDC secure access into the installation, and the components inside it rely on that boundary together with the credentials described in this chapter.

4.3 How the layers fit together

In a full installation the browser, the proxy, the Web Apps server, the Mosart servers and the identity provider are all involved, and the settings are spread across them:

1. The browser signs in against the identity provider, through the Web Apps server, and receives a session cookie.
2. The Web Apps server reads the **ID token** to decide whether an app may open, and which controls that user sees.
3. For calls that reach a Mosart server, the app fetches the **access token** and sends it as a bearer token on REST, or as an `access_token` query parameter on the SignalR hub.
4. The Mosart server validates that access token against the same authority, and reads the role from it.

Security is therefore configured in two places, separately:

- **The Mosart servers**, main and backup, in Mosart central settings under *Remote Controller Service > Security*. Described in this chapter, under Securing the Mosart Server.
- **The Mosart Web Apps**, in the Web Apps Configuration Tool. Described in the *Viz Mosart Web Apps Guide* at the Vizrt Documentation Center, section *Configuration Tool*.

Enabling one without the other leaves you half secured, and some combinations do not work at all. Three rules apply throughout:

1. **Authentication requires HTTPS.** You cannot enable sign-in over plain HTTP.
2. **Authentication requires your own certificate.** The built-in certificate authority in the Mosart Web Proxy keeps the components behind it on plain HTTP, so the Configuration Tool only offers the Authentication section in custom certificate mode.
3. **Mosart server settings need a service restart.** Nothing under *Remote Controller Service > Security* takes effect until you restart the Mosart Remote Control Service, and you must do it on each server.

4.4 Roles across Viz Mosart

Roles are assigned in your identity provider, never in Viz Mosart. The role names are fixed by Vizrt. The groups you assign them to are yours.

Role pair	Applies to	Enforced by
`mosart-admin`, `mosart-viewer`	Named Overlays Editor	the Mosart server, on its REST API

Role pair	Applies to	Enforced by
`showmaker-admin`, `showmaker-viewer`	Showmaker	the Mosart Web Apps server
`smartpanel-admin`, `smartpanel-viewer`	Smart Panel, including the Timing Display API	the Mosart Web Apps server

In each pair, `-admin` grants full control and `-viewer` grants read-only access. What the identity provider must put in the token, and how to configure it, is described in Identity Provider and Certificates.

Note: Assign roles before you enable authentication. A signed-in user with no role for an app does not get a read-only view, they get an unauthorized-access screen. An identity provider with no Mosart roles configured locks everybody out, which reads as a fault rather than as missing configuration.

Info: Roles are only enforced while authentication is enabled. With authentication off, every user is treated as an administrator. On the Mosart server, role enforcement currently covers the Named Overlays API; the playout surfaces accept any signed-in user, and role coverage there is tracked separately.

4.5 Setup at a glance

Four steps, in this order. The first two are usually done by your IT department, the last two in Viz Mosart.

Step	Who	What	Described in
1. Identity provider	IT	Create the OIDC client, define the role names, and make sure the roles reach both tokens.	Identity Provider and Certificates
2. Certificates	IT	One PEM certificate with an RSA key, covering the Web Apps host and both Mosart servers.	Identity Provider and Certificates
3. Mosart Web Apps	Mosart administrator	HTTPS with that certificate, then authentication, in the Configuration Tool.	the *Viz Mosart Web Apps Guide*, section *Configuration Tool*, at the Vizrt Documentation Center; the essentials are summarized below

Step	Who	What	Described in
4. Mosart servers	Mosart administrator	HTTPS and OIDC on the main and the backup server, then restart the service on each.	Securing the Mosart Server

The Web Apps half is configured in the Web Apps Configuration Tool and documented in the *Viz Mosart Web Apps Guide* at the Vizrt Documentation Center, section *Configuration Tool*. Three things there matter to the rest of this chapter:

1. In **Server Configuration**, HTTPS must use **Custom certificate** mode with your own certificate; the Authentication section only appears in that mode. Enter the Authority URL, Client ID and Client secret from step 1.
2. On the **Studios** page, confirm the Mosart main and backup server ports read **55168** once the Mosart servers are on HTTPS.
3. Add `offline_access` in the **Scopes** field. Without it an unattended screen, such as a Timing Display on the studio wall, ends up on a login page some hours after sign-in, once the session expires.

4.6 End to end verification

When all four steps are done:

1. **Sign-in works.** Open any Mosart web app. You are redirected to your identity provider and returned signed in.
2. **The access token carries the roles.** Do this before testing roles at all. Signed in, open the browser console and run:

   ```
   fetch('/auth/token').then(r => r.json()).then(d =>
   console.log(JSON.parse(atob(d.accessToken.split('.')[1]))))
   ```

   Look for a flat `"roles": [...]` array, and check that `aud` matches the OIDC audience configured on the Mosart server. If the roles are missing here but the app opened correctly, they are in the ID token only. Fix that before going on.
3. **A viewer can read but not write.** As a user holding only `mosart-viewer`, the Named Overlays Editor opens read-only, and a direct write to the named overlays API returns 403.
4. **An administrator can write.** The same operation as `mosart-admin` succeeds.
5. **Repeat against the backup server.** It is configured separately, so test it separately.

4.7 Troubleshooting

Symptom	Cause
403 on writes, 200 on reads	Correct behaviour for a viewer. The feature is working.

Symptom	Cause
403 on reads as well, for a user who has a role	The role is not reaching the Mosart server. Check step 2 of Check it end to end: the roles are probably in the ID token only.
401 on every call, with a token that looks valid	The OIDC audience does not match the token's `aud`, or the Mosart Remote Control Service was not restarted after the settings changed.
401 from an integration that used to work	Expected, if it authenticates with the REST API key. See Choosing an Authentication Mode.
With authentication enabled, every call to a server returns 500	The Mosart server cannot reach the identity provider. The usual reasons are wrong DNS, a firewall, or an identity provider certificate that is not trusted on that machine. To check, open your Authority URL with `/.well-known/openid-configuration` appended in a browser on the Mosart server machine. Every OpenID Connect provider answers on that address. When it returns JSON with no certificate warning, the connection is good. Restart the Mosart Remote Control Service afterwards. An invalid token gives 401, never 500.
A signed-in user sees an unauthorized-access screen	That user holds no role for the app. Assign one in the identity provider.
Everything works until a failover	The backup server was not configured, or not restarted.
A Mosart server, main or backup, is red in all web apps, while the desktop applications on that machine look normal	The service cannot load the HTTPS certificate, so its web port never starts. Often the certificate paths are wrong on that particular machine, for example when settings are copied from another server and the paths do not exist locally. Check the Mosart server log for "Failed to load the certificate". Correct the paths under Remote Controller Service > Security and restart the service on that machine.
Certificate warnings, or a blank embedded plugin	The certificate is not trusted on that machine, or its host names do not match what was entered.
503 or 504 from a studio route	Not a security problem. No Mosart server in the pair is currently on air.

Copyright © 2026 Vizrt

Symptom	Cause
The Mosart server settings page opened from the Configuration Tool asks for sign-in, or reports that the administrator role is required	Expected while OIDC is enabled: the web settings editor has no browser sign-in. Edit the settings on that server from a Viz Mosart desktop application instead: in Manus Administrator type *settings* in the console, or in AV Automation use **Devices > Settings**.

4.8 Choosing an Authentication Mode

A Mosart server runs in one of three modes. They are mutually exclusive, and the choice is a trade-off rather than a progression.

Mode	What guards the REST API	Per-user identity	Roles enforced by the Mosart server	API-key clients
Default, out of the box	nothing	no	no	not applicable
API key	one shared secret, sent as `X-API-Key`	no	no	yes
OIDC authentication	a token issued by your identity provider	yes	named overlays only, today	**no**

The Roles column is for the Mosart server only. is about the Mosart server only. The Mosart Web Apps enforce their own roles independently: Showmaker and Smart Panel gate their APIs on the Web Apps server whether or not the Mosart server uses OIDC. On the Mosart server, the playout surfaces, meaning the SignalR hub and the `timeline`, `fader`, `devices` and `settings` APIs, accept any signed-in user, so a user holding no Mosart role can still take things on air.

The last column is what usually decides the mode. **Enabling OIDC turns the API key off**, so a single Mosart server cannot serve a signed-in browser user and an API-key integration at the same time.

Which to choose:

- **API key** if you have integrations that call the Mosart REST API directly and cannot obtain a token: the Elgato Stream Deck plugin, or your own scripts without an OAuth flow. You keep those integrations and give up role enforcement. A Generic REST device no longer forces this choice, since it can sign in with the Client credentials grant.
- **OIDC** if what matters is that the people using the web apps are identified, and that a viewer cannot change named overlays by calling the API directly.

4.8.1 Do you need the Mosart Web Apps?

It depends on which part of this chapter you are following.

- **For HTTPS and the API key, no.** Both are settings on the Mosart server, and they work with no Web Apps installed.
- **For OIDC sign-in, yes in practice.** The Mosart Web Apps are what sign the user in, hold the session, and pass the resulting access token to the browser apps.

The Mosart server itself is not fussy about where a token comes from. It validates any token issued by the configured authority, and needs nothing from the Web Apps to do that. In a server-only installation, something else has to obtain the token: a Generic REST device using the Client credentials grant can, and so can your own integrations that perform an OAuth flow. Browser sign-in always needs the Web Apps.

Copyright © 2026 Vizrt

4.8.2 Third-party integrations and the REST API

Anything that calls the Mosart REST API is affected by the mode you choose.

With the API key. The integration sends `X-API-Key` with the value from *Remote Controller Service > Security*. Simple, and it works from anywhere that can reach the server. The key identifies no user, carries no role, and is the same key for every caller, so nothing in the audit trail says who did what.

With OIDC. The integration has to obtain an access token from your identity provider and send it as `Authorization: Bearer <token>`. The Mosart server validates it against the configured authority. To create, change or delete named overlays, that token must carry `mosart-admin` in its `roles` claim, exactly as a user's token does. An integration that cannot perform an OAuth flow cannot use this mode.

For health checks, use `GET /status`, which needs no credential in either mode.

Known integration limits

Integration	With the API key	With OIDC
Your own scripts and panels	Works	Works, if they can obtain a token
Elgato Stream Deck plugin	Works	**Not yet.** It authenticates with an API key only and has no sign-in flow.
Generic REST device pointed at Mosart's own REST API	Works	**Works, from Viz Mosart 5.16.** Set the device's authentication type to **OAuth2** with the **Client credentials** grant, so it signs in as its own service account. See the example on Generic REST Drivers. Earlier versions offered only the API key, or a password grant that an OIDC provider will not normally accept.

The Stream Deck plugin is a Vizrt component, so this is a current client limitation rather than a design decision. The Mosart server itself accepts a valid token from any caller; what the plugin lacks is a way to obtain one.

> **Info**: A device token carries the roles assigned to its service account, not a person's. For a Generic REST device to change named overlays, IT assigns `mosart-admin` to the client created for it, as described in Identity Provider and Certificates.

> **Note**: The Stream Deck plugin also requires the Mosart Web Apps. Its **Mosart Url** setting points at the Web Apps host, and it reaches the Mosart server through the proxy rather than directly, so it is not an option for a server-only installation.

> **Info**: A Generic REST device that calls a **third-party** service is unaffected by any of this. The limitation applies only when Viz Mosart controls Viz Mosart through its own REST API.

Copyright © 2026 Vizrt

4.8.3 What works when OIDC is enabled

Feature or integration	With OIDC enabled
Named Overlays Editor	Works, with roles enforced.
Status endpoints, `/status` and `/build`	Work, and need no credential. Monitoring keeps working.
Mosart SignalR hub	Works. The token is passed as an `access_token` query parameter.
Playout REST API (`timeline`, `fader`, `devices`, `settings`)	Works for any signed-in user. No role check.
Any client sending the REST API key	**Refused with 401.** The key is ignored while OIDC is on.
A Generic REST device pointed at Mosart's own REST API	Works, with the **Client credentials** grant, from Viz Mosart 5.16.
Elgato Stream Deck plugin	**Not yet**, apart from its status check.

4.9 Identity Provider and Certificates

What your IT department prepares before anything is configured in Viz Mosart: the OIDC client, the roles, and the certificate. At the end of this page, IT hands back three values: the Authority (issuer) URL, the Client ID and the Client Secret.

4.9.1 The identity provider

Any standards-compliant OpenID Connect provider works. Keycloak, Okta and Microsoft Entra ID are the ones documented and tested. Your IT department needs to:

1. Create an OIDC client, or app registration, for the Mosart Web Apps, using the authorization code flow with a client secret, and register these URIs on the Web Apps host:

Purpose	URI
Redirect	`https://<webapps-host>/signin-oidc`
Post-logout redirect	`https://<webapps-host>/signout-callback-oidc`
Provider-initiated logout, optional	`https://<webapps-host>/signout-oidc`

2. Define roles with exactly these names, for the apps you use:
   - `mosart-admin`, `mosart-viewer` for the Named Overlays Editor
   - `showmaker-admin`, `showmaker-viewer` for Showmaker
   - `smartpanel-admin`, `smartpanel-viewer` for Smart Panel and the Timing Display
3. Make sure the roles reach the tokens as described in The roles claim below.
4. Assign the roles to your users or groups.
5. Allow the scopes the Web Apps request, as described in The scopes below.
6. If a Generic REST device will call the Mosart REST API, create a separate client for it, using the client credentials flow, and assign the roles that device needs (for example `mosart-admin` to manage named overlays) to that client's service account. A device token carries the service account's roles, not a user's. Hand its Client ID and secret to whoever configures the device.
7. Hand back three values:

Item	Used by
Authority, or issuer, URL	both the Mosart servers and the Web Apps
Client ID	the Web Apps, and the audience value on the Mosart server
Client secret	the Web Apps only

4.9.2 The scopes

The Web Apps always request `openid`, `profile` and `email`. Anything else is added in the **Scopes** field of the Web Apps Configuration Tool, and the client in your identity provider has to allow it.

One scope is worth adding deliberately: **`offline_access`**.

A sign-in gives the Web Apps tokens that expire, typically within an hour, and the session itself has a lifetime too. Someone working in an app never notices, because their activity keeps the session alive, and if it ever ends they simply sign in again. A screen that nobody is sitting at is different: a Rundown Viewer or Smart Panel on a gallery wall has no one to sign it back in, so when its session ends it is redirected to the provider's sign-in page and stays there, showing a login form for the rest of the day.

Requesting `offline_access` asks the identity provider for a *refresh token*, which lets the Web Apps server obtain fresh tokens in the background, with no user present. Add it whenever any screen runs unattended.

> **Info**: Most providers allow `offline_access` by default, but some require it to be enabled per client. If unattended screens still end up on a sign-in page after a few hours, check that the provider is actually issuing a refresh token for this client.

4.9.3 The roles claim

The claim carrying the roles must be:

- **flat and top-level**, not nested. Keycloak's default `realm_access.roles` is not read, because the claim name is matched exactly. Add a protocol mapper that writes a top-level `roles` claim.
- **named exactly** `roles`.
- **present in both the ID token and the access token.**

That last point is the most common misconfiguration in a new installation:

Layer	Reads	Decides
Mosart Web Apps server	the ID token	whether an app opens, and which controls appear
Mosart server	the access token	whether the REST API accepts the call

If the roles reach only the ID token, the apps look completely correct and every save fails. In Keycloak, the role mapper must be enabled for the ID token and the access token separately. In Microsoft Entra ID, app roles that you have defined and assigned are emitted in a `roles` claim with no further claim mapping.

For a provider-specific walkthrough, see Vizrt SSO (Keycloak) in the Appendix.

4.9.4 The certificate

Authentication requires HTTPS with your own certificate; the built-in one is not enough. One certificate covering every host that presents it is simplest. Ask IT for:

- **Type**: X.509.
- **Format**: PEM-encoded certificate and private key. A `.pfx` must be converted to PEM first.
- **Key type**: RSA. ECDSA is not supported.
- **Encryption**: the private key must be encrypted and supplied with its password. An encrypted key begins with `-----BEGIN ENCRYPTED PRIVATE KEY-----`.
- **Host coverage**: the Subject Alternative Names must cover the Web Apps host and both Mosart servers. If the certificate names hosts by fully qualified domain name, the addresses you enter in Mosart and in the Configuration Tool must be fully qualified too. The comparison is case sensitive.

4.9.5 Trust the certificate everywhere

The certificate must be trusted on every machine involved: the Web Apps and proxy host, each Mosart server, and each machine running a browser client.

An untrusted certificate normally shows as a browser warning. When a Mosart web app is embedded as a plugin inside another application, such as ENPS, there may be no warning at all and the page simply stays blank.

Trust also runs the other way. Each Mosart server connects to the identity provider to validate sign-ins, and it checks the provider's certificate the same way a browser does. So the identity provider's certificate must be trusted on both Mosart servers, and the authority host name must resolve correctly from each of them.

Copyright © 2026 Vizrt

4.10 Securing the Mosart Server

Everything on this page is done on the Mosart server, in Mosart central settings, under **Remote Controller Service > Security**. Open those settings from Manus Administrator, AV Automation or Media Admin.

> **Note**: Do all of it on the main server **and** on the backup server. The backup is configured separately and is the easiest thing to leave half-done. A setup that works until a failover is almost always a backup server that was never configured, or never restarted.

> **Info**: Two similar names are in play. *Remote Controller Service* is the settings category you navigate to. The Windows service you restart is listed as `MosartRemoteControlService`.

4.10.1 Turn on HTTPS

Under the **HTTPS** section:

Setting	Value
Use HTTPS	on
Certificate path	path to your `.crt` file
Certificate key path	path to your `.key` file
Certificate password	the private key password
SignalR HTTPS port	55168 (the default)

The HTTPS port is the plain port plus one. Plain HTTP is 55167, so HTTPS is 55168.

Restart the Mosart Remote Control Service, then confirm the server answers on the new port:

```
https://<mosart-server>:55168/api/v1/status
```

You should get a status document, with no certificate warning.

4.10.2 Turn on OIDC authentication

Under the **OIDC / JWT authentication** section:

Setting	Value
Use OIDC authentication	on. Requires HTTPS.

Setting	Value
OIDC authority	the Authority URL, the same one the Web Apps use
OIDC audience	the Client ID, or leave empty to skip audience validation

Restart the Mosart Remote Control Service.

> **Note**: If the audience does not match the `aud` value in the token, every call is refused with 401 even though the token itself is valid. Leave the audience empty if you are unsure, and set it once sign-in is working.

> **Note**: Turning this on switches off the REST API key. The key is not checked at all, so every integration that authenticates to the Mosart REST API with the REST API key is refused with 401. See Choosing an Authentication Mode, and plan for this before you enable authentication in production.

4.10.3 Roles the Mosart server enforces

With OIDC on, the Mosart server reads the user's role from the access token and enforces it on the Named Overlays REST API:

Role	On the Named Overlays API
`mosart-admin`	Read and write: create, edit, delete and import overlays.
`mosart-viewer`	Read only. Writes are refused with 403.

A signed-in user holding neither role is refused on reads as well as writes.

> **Info**: Role enforcement on the Mosart server requires Viz Mosart 5.16 or later together with Mosart Web Apps 3.2 or later. In earlier versions the Mosart server checked only that a caller was authenticated.

Other surfaces on the Mosart server accept any signed-in user, whatever role they hold. That includes the SignalR hub and the playout REST API (`timeline` , `fader` , `devices` and `settings`). Role coverage for those is tracked separately.

4.10.4 Check the server on its own

Before involving the browser, confirm the server behaves:

1. **It answers on HTTPS.** `https://<mosart-server>:55168/api/v1/status` returns a status document.
2. **It refuses anonymous callers.** The same URL under `/api/v1/namedoverlays` , with no credential, returns 401.

3. **It refuses the API key.** With OIDC on, a call presenting `X-API-Key` also returns 401. That is correct, and it is the check that tells you OIDC really took effect.

Then continue with the end-to-end check on the Security chapter page.

4.11 Password-Protected Features

Viz Mosart has several individually password-protected features. These are independent of HTTPS and OIDC, and apply whether or not authentication is enabled. On the desktop applications, they are the main access control alongside network isolation.

4.11.1 Template Editor password

Password protection can be enabled for the Template Editor. When enabled, the user is prompted for the password when opening the Template Editor. By default no protection is enabled.

See Setup and Administration of Template Editor and Configuration Issues.

4.11.2 GPO

Authentication to a GPO password-protected connection is set in AV Automation under **Devices > Properties > GPI/O**.

See GPI/O.

4.11.3 Generic REST driver credentials

When Viz Mosart calls an external REST service, the credentials it presents are configured on the device. The available authentication types are described in Authentication types on the Generic REST Drivers page.

> **Info**: This is Viz Mosart acting as a client. For securing Mosart's own REST API, see Securing the Mosart Server in this chapter.

4.11.4 Vizrt Graphics MSE database

User: username on the Oracle database. Default: *pilot*.

Password: password on the Oracle database. Default: *pilot*.

Datasource: TNS name or connection string of the Oracle database. Default: *vizrtdb*.

Schema: Oracle database schema for the Viz Pilot connection. Default: *PILOT*.

See Graphics (Vizrt Settings).

4.11.5 TriCaster

TriCaster API credentials are required, and are initially set to **admin** / **admin**.

> **Note**: Contact the TriCaster administrator, or sign in to the TriCaster Administration panel, to manage these values.

See To connect TriCaster to Viz Mosart.

Copyright © 2026 Vizrt

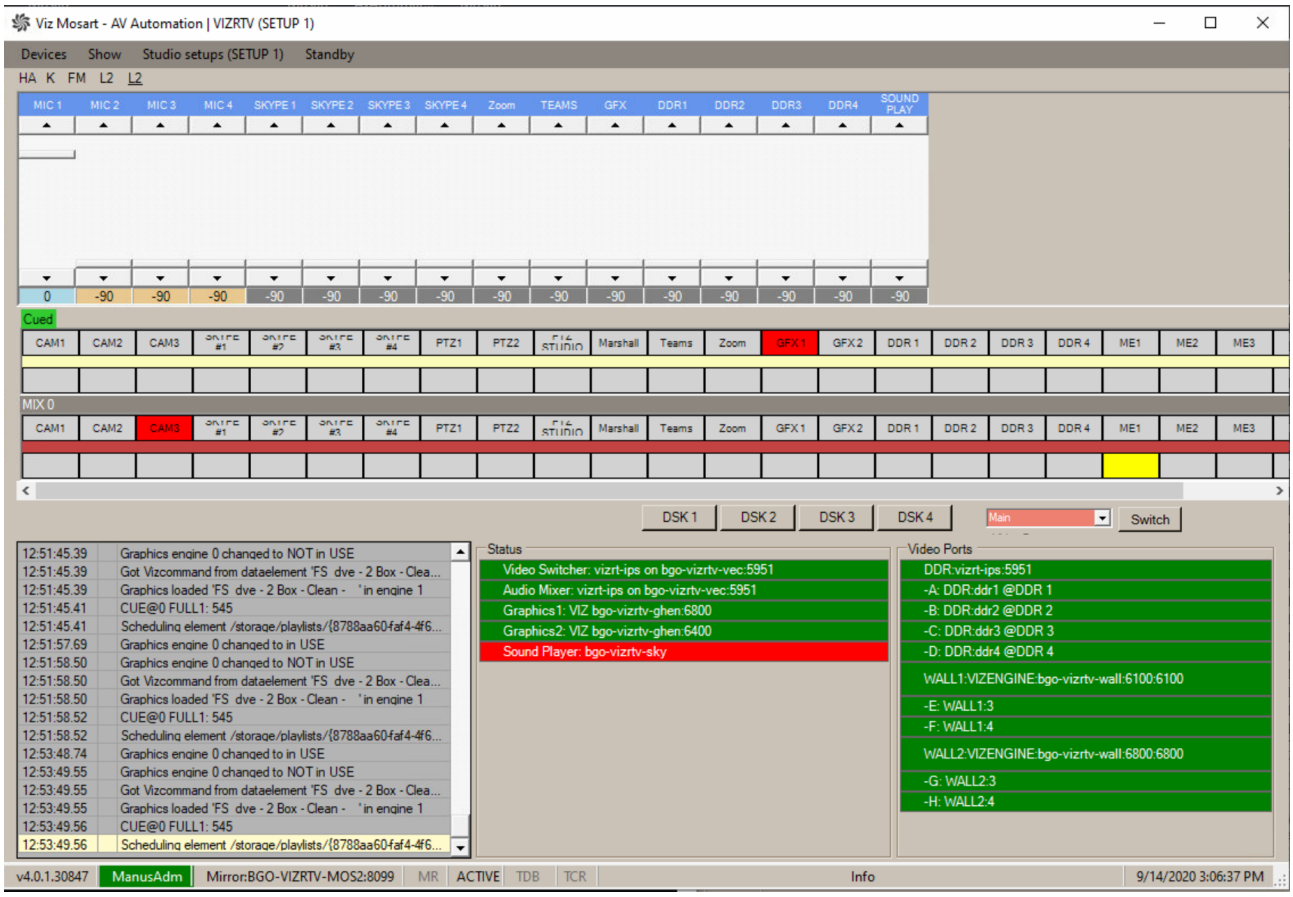
5 AV Automation

AV Automation (program file *MMAVAutomation*) forms the bridge between Viz Mosart and the devices (for example a vision mixer or video server) in the studio. It also controls full-screen graphics (but *not* Overlay Graphics engines), which from a Viz Mosart perspective, are treated as *clips*.

The following AV Automation topics are presented:

- Overview
- Menu Selections
- Logging and Status

5.1 Overview



5.1.1 AV Automation Tasks

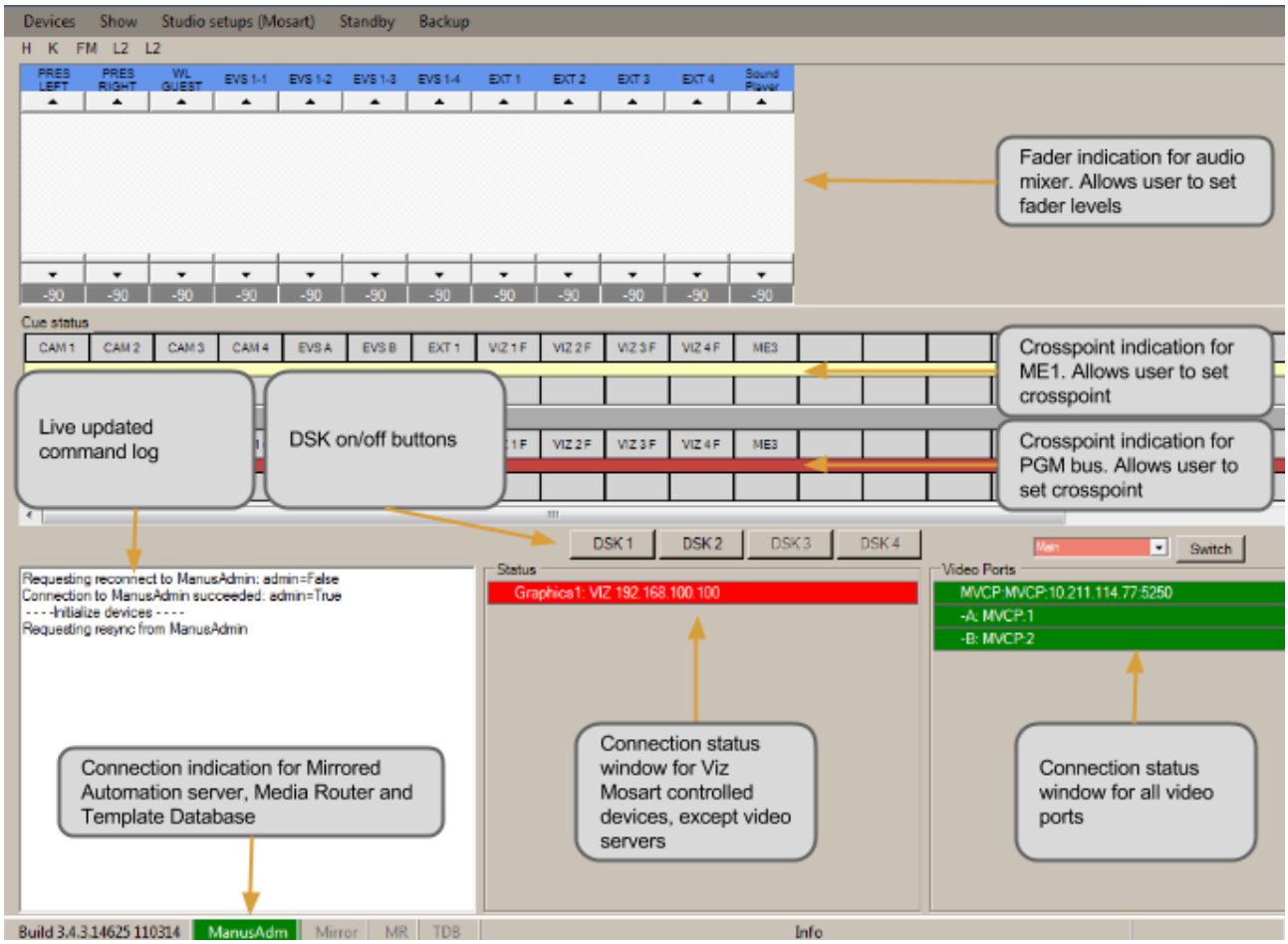
The principal tasks for AV Automation are:

- To *Cue* a clip (prepare) when in preview

- To *Play* a clip when taken on air
- To *Manage control commands* to studio devices.

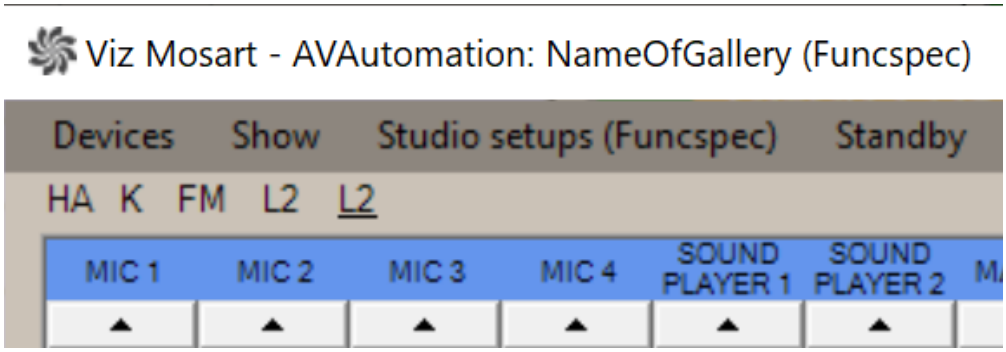
5.1.2 AV Automation Characteristics

- AV Automation is the device handler of Viz Mosart. It communicates with all connected devices such as cameras, video servers and vision mixers.
- All device drivers are stored in AV Automation, offering the administrator a single point of configuration.
- All device commands are stored as predefined Viz Mosart *templates*. They are managed in the Template Editor, and saved in `C:\channeltemplates` or in a separate template database.
- As well as providing tools for configuring these devices, it includes Template Editor, for defining and customizing templates.
- On the fly commands can then be issued directly to each device through the Viz Mosart GUI, or as pre-programmed operations driven by the rundown of a connected NRCS.



- The **Switch** button switches video server salvo. The drop-down menu to the left of **Switch**, lists the available video server salvos.

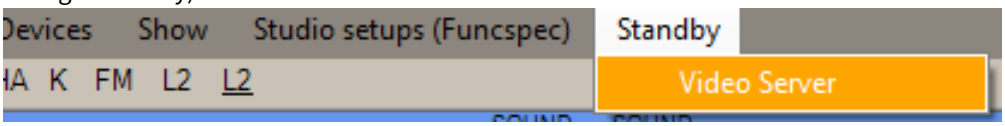
5.2 Menu Selections



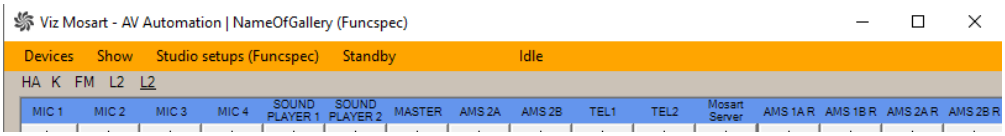
- **Devices**:
  - **Preferences**: Opens AV Automation Device Properties.
  - **Settings**: Opens Mosart Administrator Settings.
  - **Template editor**: Opens Template Editor.
  - **A/V Setup**: Opens Audio and Video Setup.
  - **Record**: See Show Recordings.
  - **Change password**: Creates or changes the Template Editor Password.
- **Show**:
  - **Show MVCP Control**: Deprecated.
  - **Instrumentation Panel**: Opens the Mosart Instrumentation Panel.
- **Studio setups**: Selects the template set to use for the current show. These template sets are *not* available for selection:
  - those marked as **Hide from user**
  - the special template set **DirectTakes**.

> **Note**: For more information on the concepts of *template set, Directtakes,* and *direct take* templates, please refer to the *Template Sets* section in the Templates chapter of the *Viz Mosart User Guide*.

- **Standby**: Lists all configured devices and their connection status (Green: connected, Red: not connected, Orange: standby).



  From this menu, a device can be placed in Standby (or taken out of Standby). (*Standby* means the device is not under control of Viz Mosart; no commands will be sent to it from Mosart.)
  The Viz Mosart operator can also perform this action from the main UI. See the **Viz Mosart User Interface** section of the *Viz Mosart User Guide*, under *Main Menu*, *Standby* for further details.
- **Idle**: This is an indicator that AV Automation is in idle mode, not a menu option. In this state, the menu bar turns orange.



As long as AV Automation is *not* in idle mode, there is no 'Idle' in the menu bar.

5.3 Logging and Status

Below the fader and switch representations are three monitoring panels.

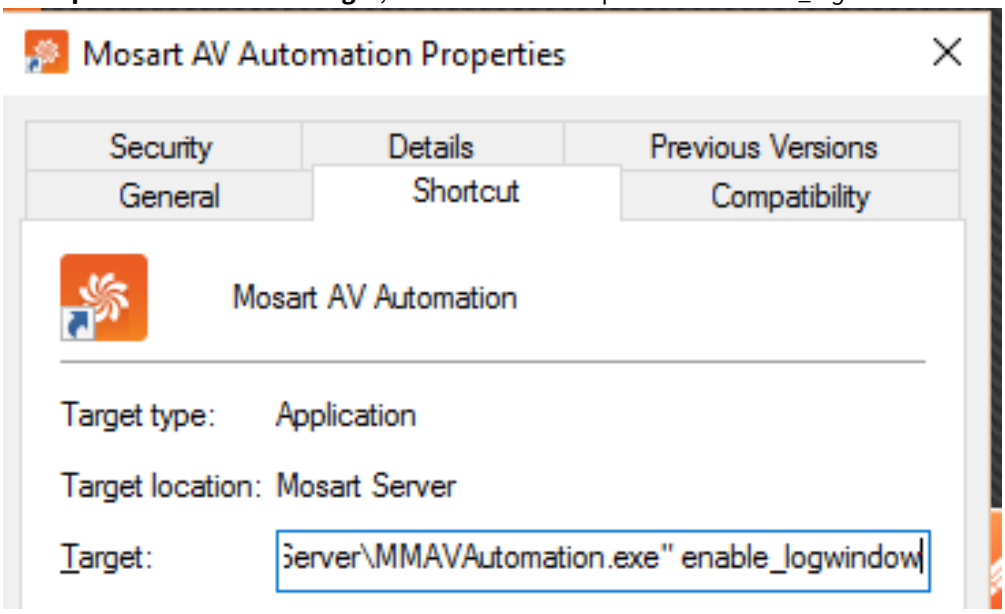
5.3.1 Logging Panel

To the left is a Logging panel with time stamped event details. Logging is disabled by default.

To Enable Logging

There are two options:

- Right-click in the panel and select **Enable**
- In **Properties > Shortcut> Target**, add the command parameter *enable_logwindow*.



Logging will then be enabled at AVAutomation start-up.

To Clear Logging

- Clear the Logging panel by double-clicking in the panel.

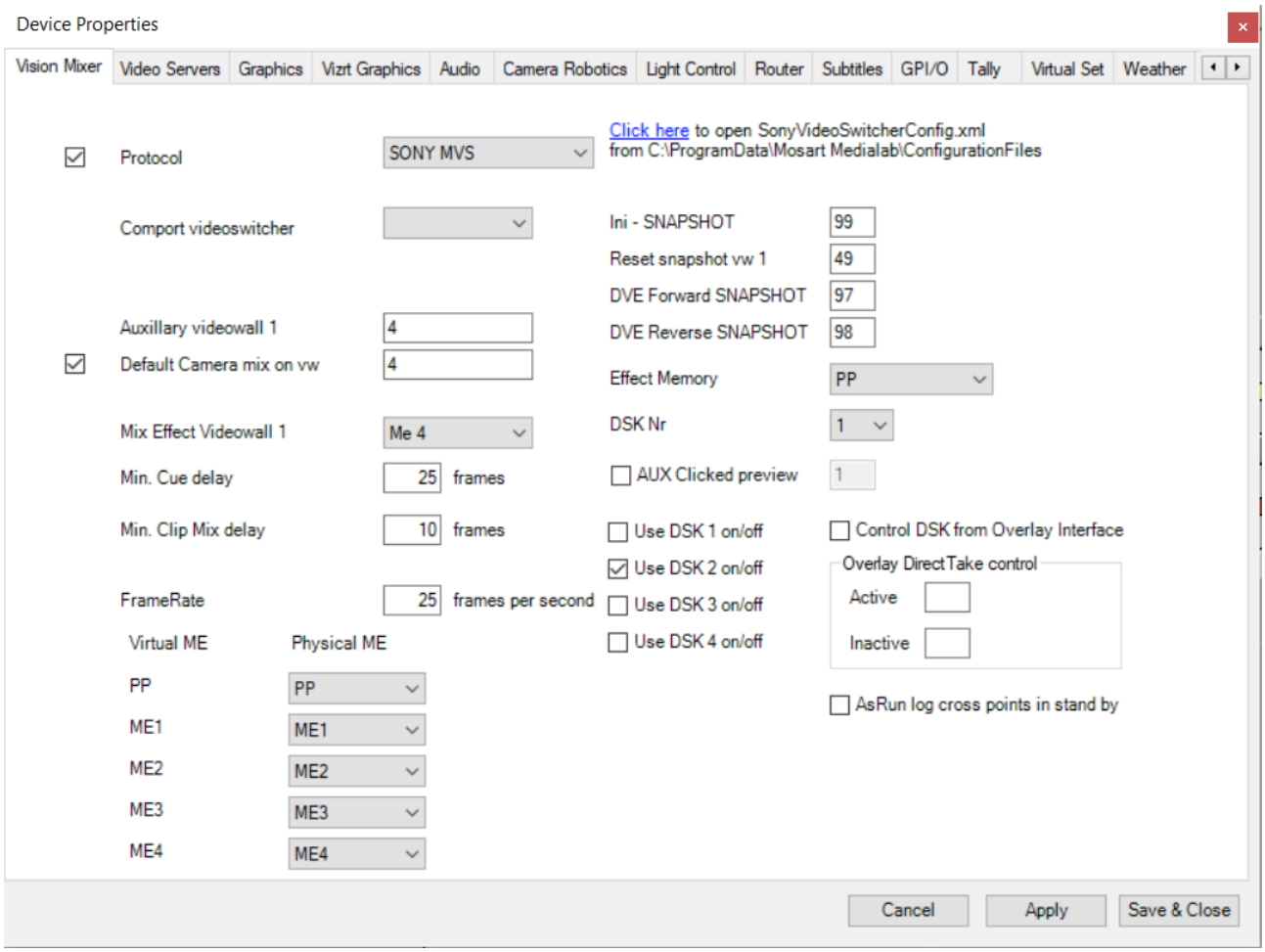
The following AV Automation topics are presented:

- AV Automation Device Properties
- A/V Setup (Audio and Video Setup Config File)
- Device Connection Strings and Configuration Files
- Template Editor
- Mosart Instrumentation Panel
- Recording a Show

Copyright © 2026 Vizrt

5.4 AV Automation Device Properties

AV Automation can send commands to all connected broadcast devices. For the commands to be received, and interpreted according to the device's specifications, the device must be configured within Viz Mosart.



- To configure device properties in AV Automation, select **Devices > Properties**.

> **Note:** AV Automation must be restarted before any changes take effect.

The Device Properties menu has three common buttons:

- **Cancel**: Exit this menu without saving any changes.
- **Apply**: Save all changes and reconnect the device on the active tab (only applies to tabs **Vision Mixer**, **Video Servers**, **Graphics**, **Audio**, **Camera Robotics**, **Light Control** and **Subtitles**).
- **Save & Close**: Save all changes and close the menu.

The following device properties are presented:

- Vision Mixer

- Video Servers
- Graphics
- Vizrt Graphics
- Audio
- Camera Robotics
- Light Control
- Router
- Subtitles
- GPI/O
- Tally
- Virtual Set
- Weather
- Video Wall
- General
- Genlock

Info: For connecting between Viz Mosart and *TriCaster*, see also: TriCaster and Viz Mosart.

5.4.1 Vision Mixer

Vision Mixer Settings

There are several settings required for Viz Mosart to effectively control a specific video mixer.

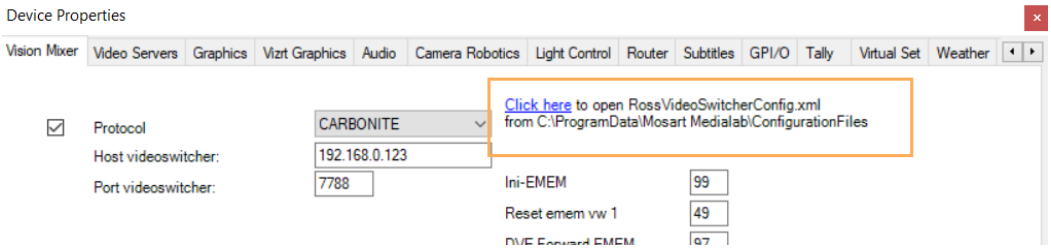
- Vision Mixer Configuration
- Usage Notes
- The Video Mixer Menu
- Common Video Mixer Connection Settings
- Moxa NPort device as the COM port

Vision Mixer Configuration

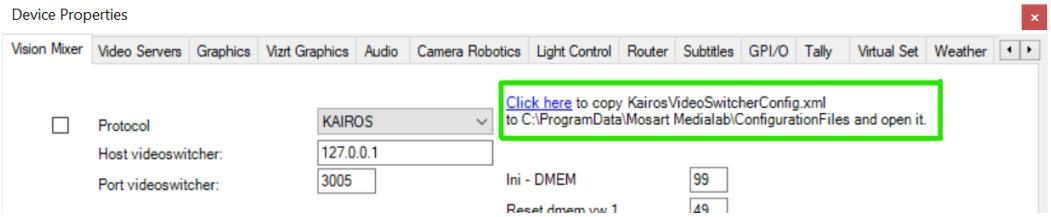
When configuring a vision mixer, the **Vision Mixer** tab in the **Device Properties** window links to any additional configuration file used by the driver.

To modify a vision mixer configuration file

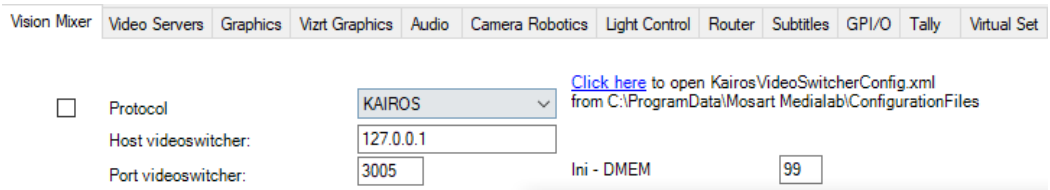
1. Open AV Automation, navigate to **Device Properties > Vision Mixer** tab **> [**Select **Protocol]**.
2. When a device is selected, either
   a. There is a corresponding configurations file that contains additional settings:
      i. This is indicated by the text *Click here to **open** xxx*.
      In the example below, a Ross settings file is already located at C:\ProgramData\Mosart Medialab\ConfigurationFiles.



      ii. Follow **Click here** to edit the configurations file.
   b. There is no corresponding configurations file:
      i. You can copy an unused default file from the Installation folder.
      This is indicated by the text *Click here to **copy** xxx*.

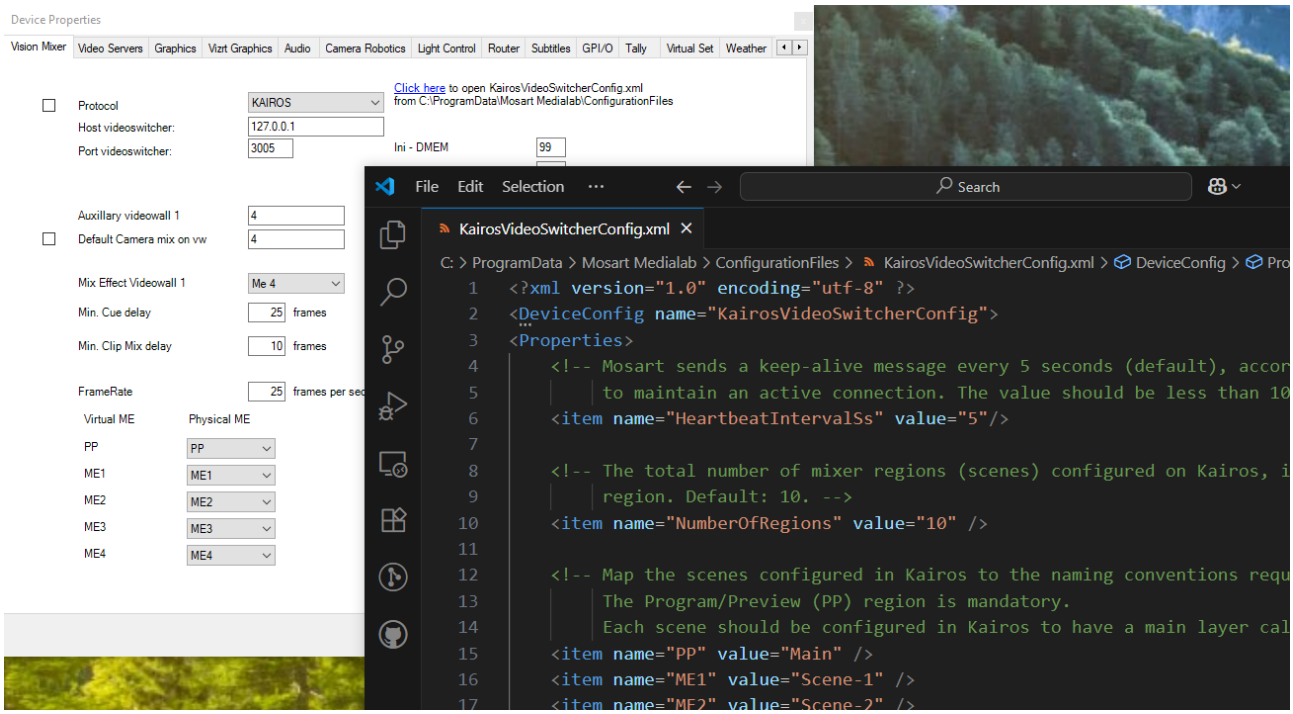


      ii. Follow **Click here**.
      iii. The information text changes to *Click here to open xxx*, as the file now exists for editing, in a standard configuration file folder.



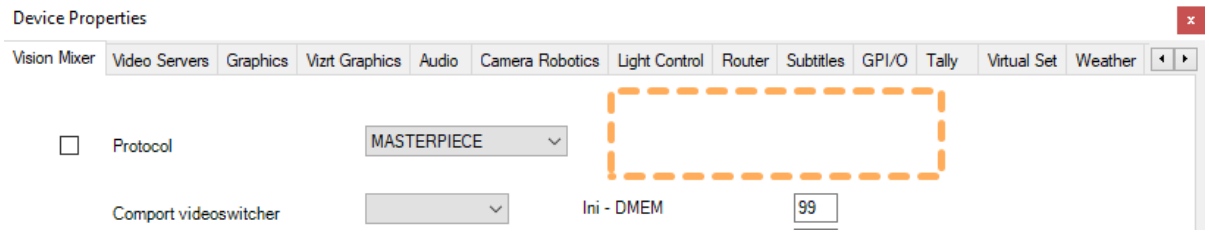
iv. Follow **Click here** to edit the configurations file.

3. The configuration file opens in your default XML file editor.



Usage Notes

- Information about any configuration file will not appear if the device does not require a configuration file.



- Configuration files are used to configure additional settings.
  See section Vision Mixer Configuration Files for examples.
- The standard configuration file folders are
  - C:\ProgramData\Mosart Medialab\ConfigurationFiles

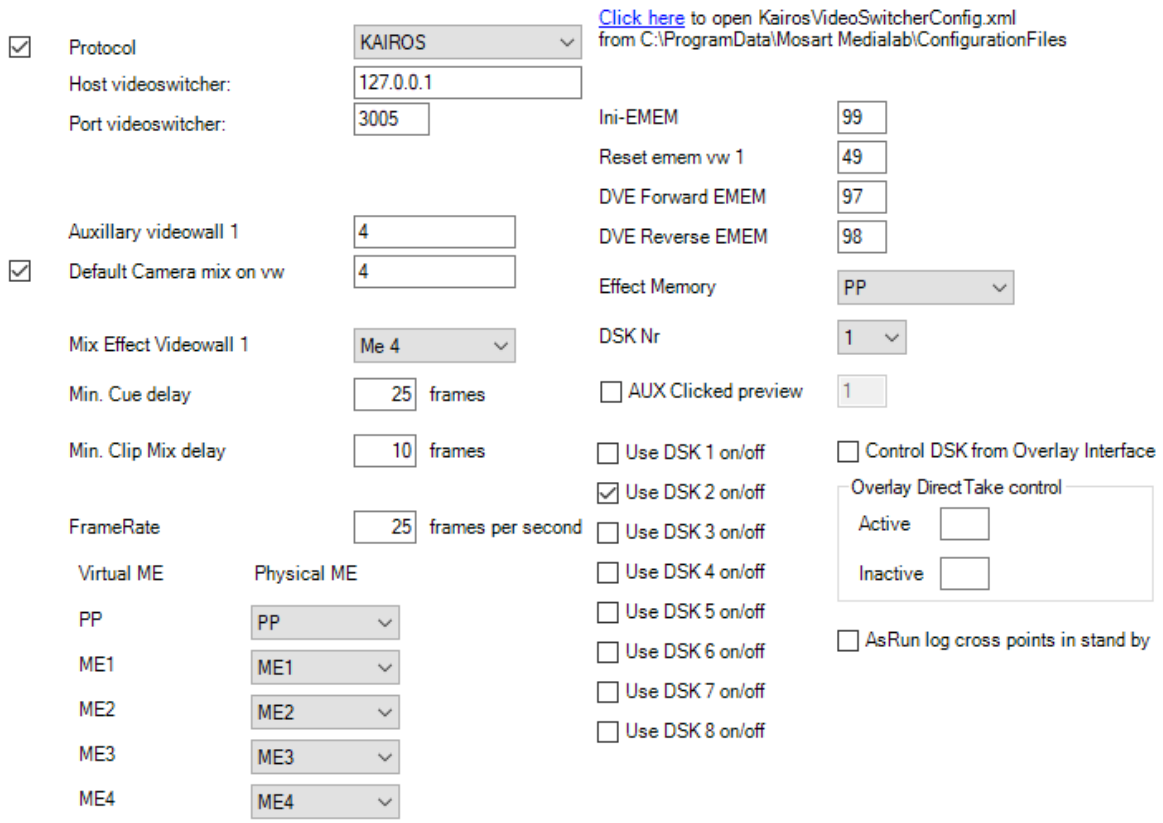
- C:\channeltemplates

If the configuration file is found in *both*
“C:\ProgramData\Mosart Medialab\ConfigurationFiles” and
“C:\channeltemplates”,
the path “C:\ProgramData\Mosart Medialab\ConfigurationFiles” is prioritized.

The Video Mixer Menu

Info:

- Most devices, especially ones controlled through IP, often have their own configuration interface (for example, accessible through a web page) where various parameters such as host/port can be inspected and modified as needed.
- Use the values found in the device configuration interface when setting their corresponding properties that are described below.



- **Protocol (check box)**: Check the box to enable video switcher control.
- **Protocol**: Select a protocol supported by your switcher:
  - ACUITY (See **Info** below)
  - ATEM
  - CARBONITE

Info

- For ACUITY and CARBONITE video switcher protocols, a *new* physical ME named *ME0* is introduced.
  ME0 shall be used to address the P/P on newer/updated Ross video switchers.
  Set this up under **AV Automation > Device Properties > Vision Mixer > [Protocol = CARBONITE|ACUITY]**.
  This ME naming can, for example, control a Ross Carbonite Ultra video switcher from Viz Mosart.
- For Carbonite, the MiniME-related settings control the corresponding MiniMEs regions on the switcher, while for Ross Acuity these settings are used to control the B-side of the corresponding mixer region.

  - GVG200: GVG/Philips/Thomson
  - GVG4000: Ross Synergy
  - GVG4000 V2
  - GV CPL
  - GV DD35
  - GV KAYENNE PRIMARY
  - GV KAYENNE SECONDARY
  - GV ZODIAK
  - KAHUNA: Snell/Grass Valley Kahuna/Kula (a dialect of SONY)
  - KAHUNA IP: Grass Valley Kahuna/Kula (a dialect of SONY)
  - KAIROS
  - MASTERPIECE
  - NOVA700: Echolab
  - SONY: Sony compatible video switchers
  - SONY BVS: Support for legacy SONY DVS/BVS 300 series
  - SONY MVS: An extension to the protocol for the SONY MVS8000
  - SONY SERIAL TALLY
  - SONY SERIAL TALLY IP
  - VIZRT ENGINE
    See also Viz Engine as a Switcher.
  - TRICASTER (was *VIZRT-IPS*)
    See also: TriCaster and Viz Mosart.
  - VMIX
- **Comport videoswitcher**: Defines the serial port the video switcher is connected to.
- **Host videoswitcher/Port videoswitcher**: Defines the IP host address and port of the video switcher.
- **Auxiliary videowall 1**: This is the auxiliary output from the video switcher to the video wall.
- **Default Camera mix on vw (check box)**: Enables the default camera in program when entering video wall mode.
- **Default Camera mix on vw**: Enter the camera number you want to use as the default camera in program when entering video wall mode.
- **Mix Effect Videowall 1**: Select the ME to use when enabling mixing in video wall mode.
- **Min. Cue delay**: Value in frames which sets the minimum delay before Viz Mosart cues the next template in preview.

- **Min. Clip Mix delay**: Value in frames which sets the minimum delay from starting the server to starting the video switcher transition.
- **FrameRate**: Specify the number of video frames per second for the system.
- **Ini-EMEM**: Initial register to recall when starting the automation. Depending on the Protocol chosen, the caption may alternatively be **Ini - DMEM** or **Ini - SNAPSHOT**.
- **Reset EMEM vw 1**: Register to recall to normalize the ME before entering video wall mode. (Depending on the Protocol chosen, the caption may alternatively be **Reset dmem vw 1** or **Reset snapshot vw 1**).
- **DVE Forward EMEM**: (not supported by Sony) EMEM to recall for running DVE forward. (Depending on the Protocol chosen, the caption may alternatively be **DVE Forward DMEM** or **DVE Forward SNAPSHOT**).
- **DVE Reverse EMEM**: (not supported by Sony) EMEM to recall for running DVE backward. (Depending on the Protocol chosen, the caption may alternatively be **DVE Reverse DMEM** or **DVE Reverse SNAPSHOT**).
- **Effect Memory**: ME to use when recalling registers for Effect use.
- **DSKNr**: Downstream keyer to use for the DSK on/off functionality.
- **AUX Clicked preview (check box and number)**: Check the box to enable AUX clicked preview, and specify the AUX bus connected to the preview monitor. When activated, it is possible to click an element in the Viz Mosart rundown, to preview the source on the monitor, via the specified AUX bus.
- **Use DSK n on/off**: Enable or disable the on/off functionality for DSK 1-N, where N depends on the protocol. Most protocols have 4 DSKs, but there could be as many as 8 DSKs.
- **Control DSK from Overlay Interface**: Check box to enable DSK to be on only when overlay graphics is present.
- **Overlay Direct Take control:**
    - **Active**: Viz Mosart will run the direct take entered here when an overlay graphic goes On Air.
    - **Inactive**: Viz Mosart will run the direct take entered here when an overlay graphic goes Off Air.
- **AsRun log cross points in stand by (check box)**: Check box to set AsRun log in stand-by.

Common Video Mixer Connection Settings

The following table presents some recommended or additional values for a range of video mixer drivers.

Driver	Settings	Configuration Interface Notes
ATEM	Host: IP address or *USB* (Default: *USB*). Port: Not used. **Info**: The connection to the ATEM switcher will always fall back to attempting a USB connection. The value *USB* serves as a visual indicator, displayed in the Status panel of AV Automation, and in logs.	ATEM Software Control must be installed on the Mosart server machine. ATEM Software Control must not be running when the Mosart driver is connected to the ATEM switcher via USB. PP and ME1 both map to physical ME1 on the ATEM switcher. PP targets downstream keyers (DSKs), while ME1 targets upstream keyers. See also: Template Device Functions.

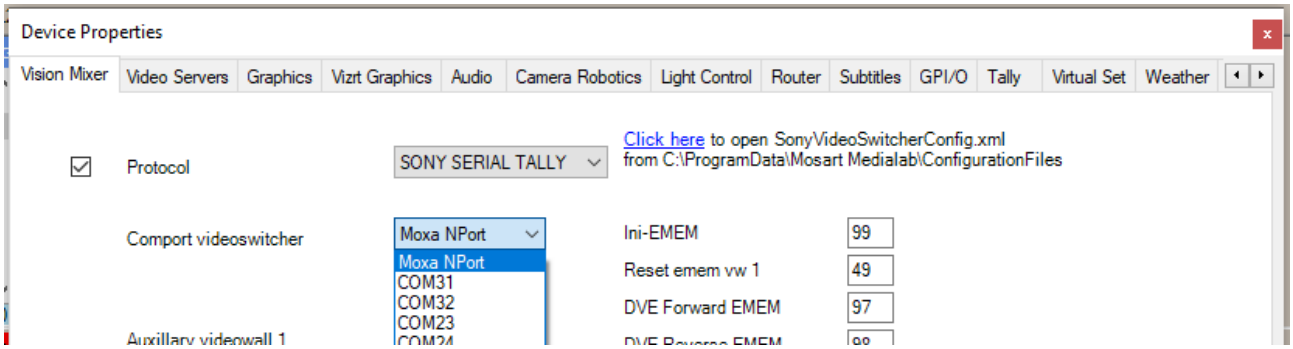
Driver	Settings	Configuration Interface Notes
GV CPL	Port video switcher: *5000* Client Port video switcher: *6040*	This driver uses UDP protocol Additional client port property used for receiving callbacks
KAIROS	Port video switcher: *3005*	
Sony	SonyVideoSwitcherConfig.xml setting <item name="ReadTimeout" value="100" />	ReadTimeout is possible to change in the *SonyVideoSwitcherConfig.xml* configuration file. See Device Configuration Files for guidelines. By default, this setting is commented out.
Sony Serial Tally	SwitcherTypeVariant (in settings, with **Ctrl+Shift+S**) should be set to *SonyCom*.	
Sony Serial Tally IP	SwitcherTypeVariant (in settings, with **Ctrl+Shift+S**) must be *cleared*. SonyVideoSwitcherConfig.xml setting <item name="ReadTimeout" value="100" />	ReadTimeout is possible to change in the SonyVideoSwitcherConfig.xml configuration file See Device Configuration Files for guidelines. By default, this setting is commented out.
VIZRT ENGINE		When working with a Viz Engine in a *switcherless*, or *Engine Switcher* configuration, refer to the specific settings in section Viz Engine as a Switcher.
TriCaster		See TriCaster and Viz Mosart.
vMix	Host: IP address or hostname of the machine running vMix (Default: *127.0.0.1*, the localhost). Port: vMix TCP API port (Default: *8099*).	No additional configuration file is required.

Moxa NPort device as the COM port

In **Comport videoswitcher** you can select *Moxa NPort* for devices using the following protocols:

- SONY
- SONY BVS
- SONY MVS

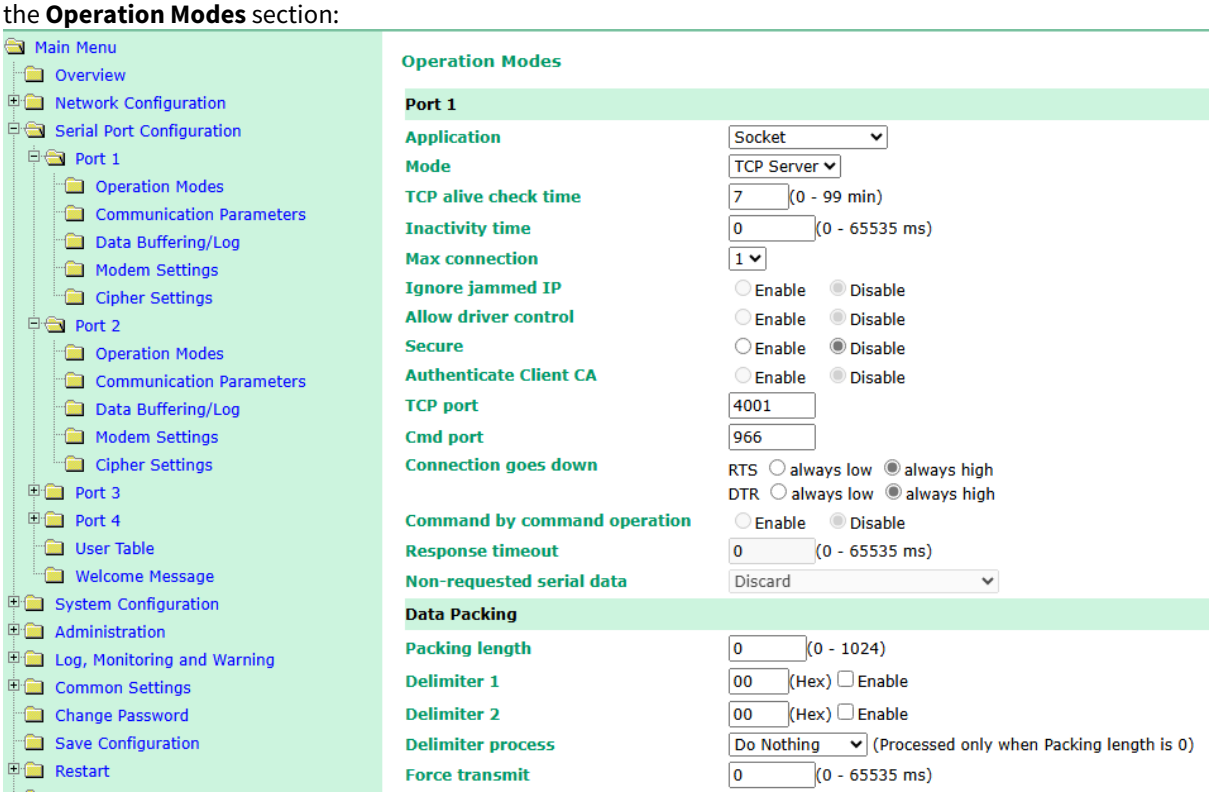
- SONY SERIAL TALLY
- KAHUNA



To use a Moxa NPort device as the COM port

Configure the Moxa NPort device:

1. Open the administration panel of the device and for the given serial port (*Port 1* in this example) navigate to the **Operation Modes** section:



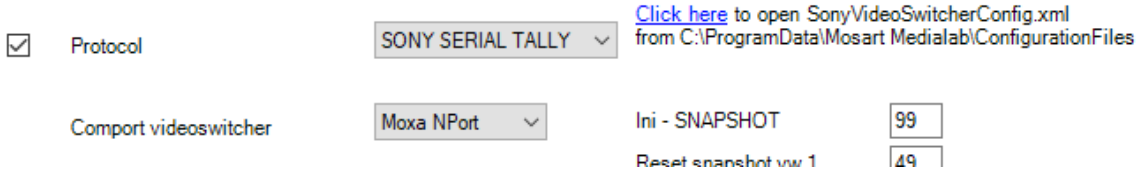
2. From the **Application** dropdown menu, select *Socket.*
3. From the **Mode** dropdown menu, select *TCP Server.*
4. Leave all other settings with their default values, but see Info below.
5. Note down:
    a. The value of **TCP port**.
    b. The value of **Cmd port**.

c. The device's **hostname/IP address**.

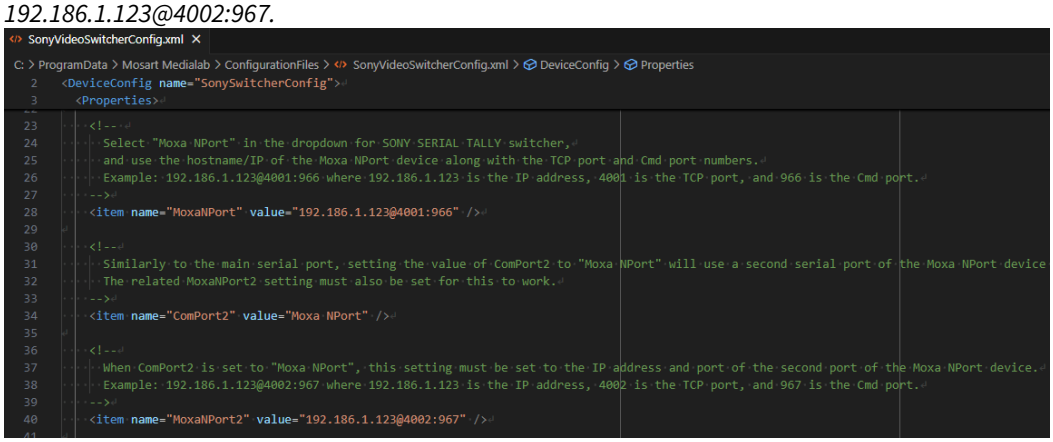
> **Info**: In the **Operation Modes** panel, if the value of **Max Connection** is more than ***1***, then set **Allow driver control** to *Enable*.
> In a setup with Viz Mosart main and backup Server using the same Moxa NPort device to connect to the vision mixer, **Max Connection** must be set to ***2***, allowing both Viz Mosart Servers to connect

Configure AV Automation:

1. From the AV Automation menubar, select **Devices > Properties > Device Properties > Vision Mixer** tab.
2. On the **Vision Mixer** tab, from the **Protocol** dropdown menu, select one of the supported protocols.
3. From **Comport videoswitcher**, select *Moxa NPort.*
4. Open the configuration file *SonyVideoSwitcherConfig.xml* by following the **Click here** link (next to **Protocol**).



   The configuration file opens in your default text editor.
5. Modify the following settings in *SonyVideoSwitcherConfig.xml*, using the values you noted down earlier:
   a. **MoxaNPort**: `value="<hostname_or_ip>@<tcp_port>:<cmd_port>"`
      i. `hostname_or_ip` is the hostname/IP address of the Moxa NPort device.
      ii. `tcp_port` and `cmd_port` are the two ports configured above for **Port 1**, for example: *192.186.1.123@4001:966*.
   b. **ComPort2**: `value="Moxa NPort"`
      i. This is used to set up the second serial port for the SCU connection, when needed by the switcher.
   c. **MoxaNPort2**: `value="<hostname_or_ip>@<tcp_port>:<cmd_port>"`
      This is used to set up a *second* serial port for the SCU connection when needed by the switcher:
      i. `hostname_or_ip` is the hostname/IP address of the Moxa NPort device
      ii. `tcp_port` and `cmd_port` are the two ports configured above for **Port 2**, for example: *192.186.1.123@4002:967.*



6. Restart AV Automation

Working status

- The Video Switcher status shall indicate that the Moxa NPort device is used for one (or both) serial ports:

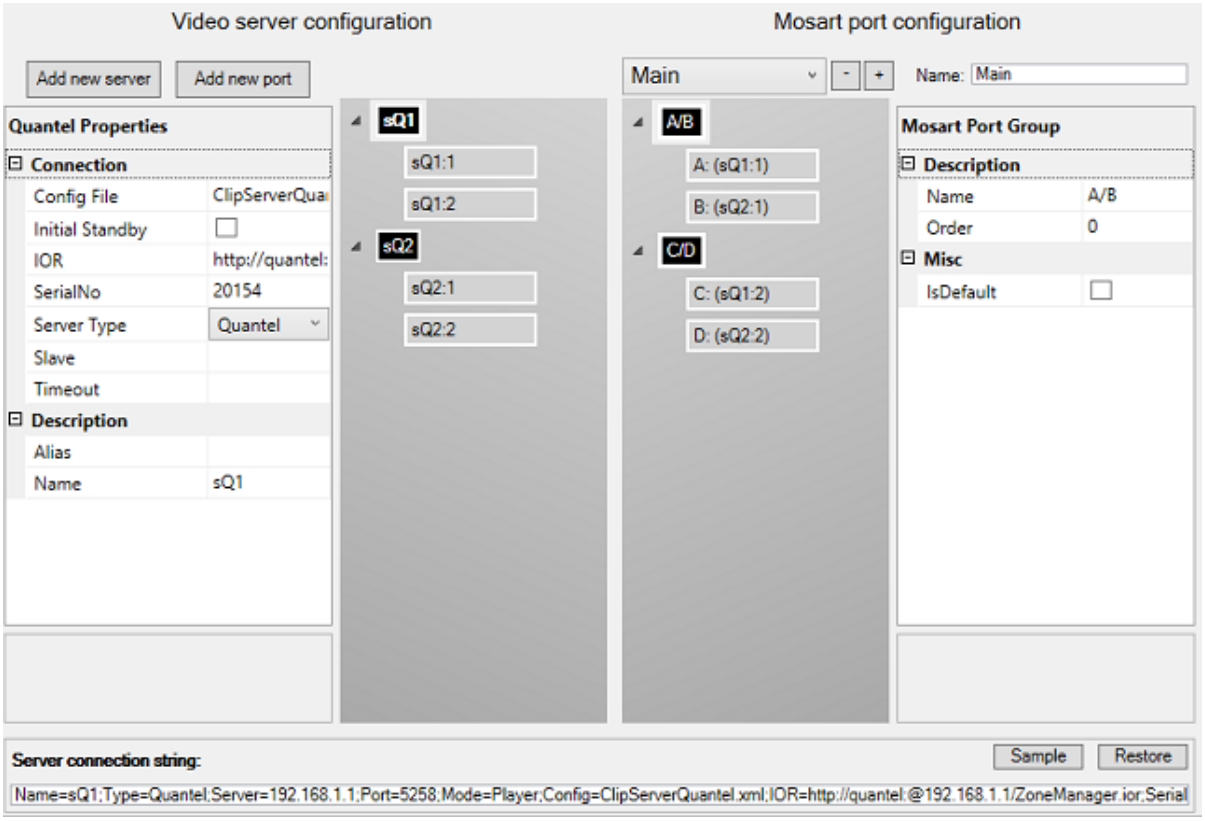
5.4.2 Video Servers

Viz Mosart is capable of controlling video servers from many different manufacturers. Although the end result is the same, each video server type operates in a slightly different way.

- Video Server Configurations
- Video Server Properties
  - Common Settings
  - Video Server Port Properties
- Working with Servers and Ports
- Mosart Port Configurations
  - Working with Mosart Port Configurations
  - Working with Salvos
- Video Server TriCaster Setup Example

It is essential that the correct settings are defined in AV Automation for Viz Mosart to effectively and correctly control your video server. You reach these settings with

- **AVAutomation > Devices > Properties> Video Servers** tab.



- The *left* side (*Video server configuration*) presents details of the physical video servers and their ports.
  The example above presents a Quantel server (as selected from the **Server Type** drop-down menu), sQ1, and one other server (sQ2), which could also be a Quantel server.
  Server sQ1 has 2 ports defined, sQ1:1 + sQ1:2.
  Server sQ2 has 2 ports defined, sQ2:1 + sQ2:2.

This is described in Working with Video Server Configurations and Working with Servers and Ports.

- The *right* side (*Mosart port configuration*) presents details of the virtual connection within Viz Mosart, with salvos, virtual ports, virtual port groups and assignments of physical server ports (as defined on the left side) to virtual ports. These concepts are presented below.
  This is described in Working with Mosart Port Configuration.

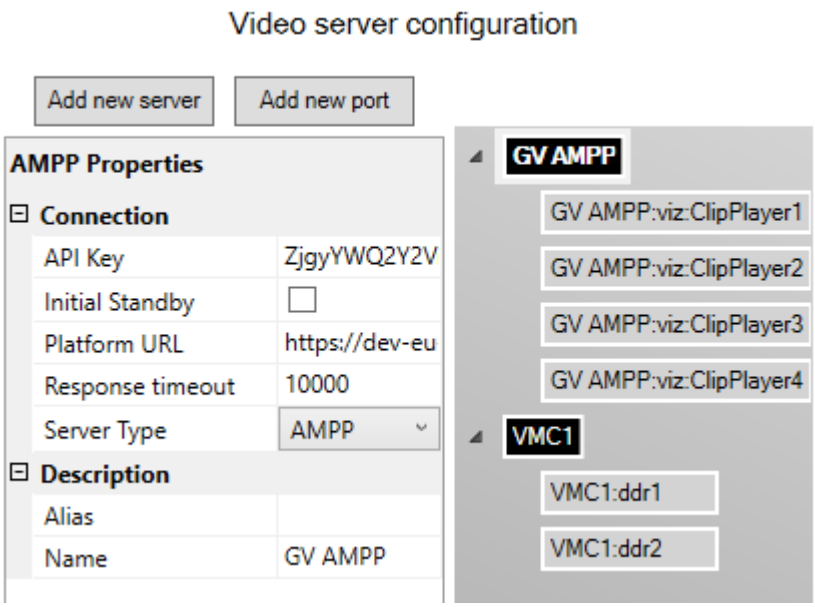
This rest of this section contains the following topics, presented according to the configuration menu illustrated above, from left to right:

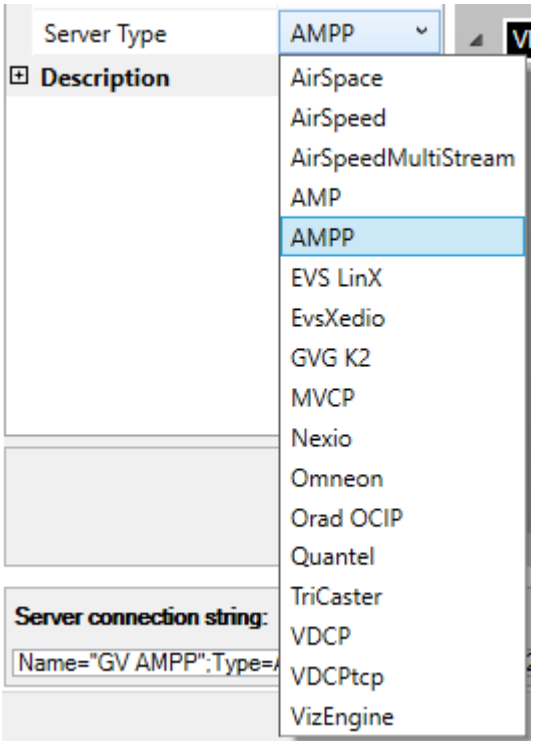
- Video Server Configurations
- Video Server Properties
  - Common Settings
  - Video Server Port Properties
- Working with Servers and Ports
- Mosart Port Configurations
  - Working with Mosart Port Configurations
  - Working with Salvos
- Video Server TriCaster Setup Example

Video Server Configurations

You can *add* or *remove* video servers to your Viz Mosart system, and *modify* the configuration of existing hardware.

Viz Mosart supports a wide range of video servers. These appear in the **Server Type** drop-down menu.





Note: Only compatible combinations of Media Administrator and AV Automation drivers can be defined. Please contact Vizrt Support for further details.

- The **Name** property links the selected video server to a Media Administrator *properties* line.
- For an explanation of the various configuration properties, please refer to the specific video server model, as presented in Video Server Properties below.
- After configuration, restart Media Administrator and AV Automation, and ensure that the GUI is connected to the required server.
- Verify that all indicator lights for the selected video server go green.

Tip: If you do not get green lights, and you see a message similar to this in the Mosart log:

```
Failed connecting to video server K2:
The source was not found, but some or all event logs could not be searched.
Inaccessible logs: Security.
```

you can try to *start AVAutomation* as *Administrator*. This is a known issue with Grass Valley servers.

Video Server Properties

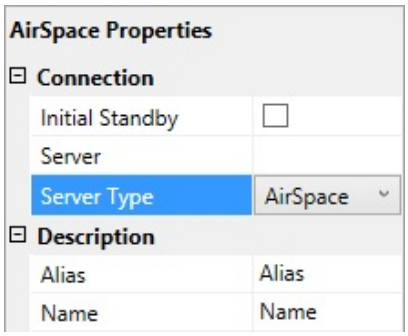
Here are some guidelines for common video servers that Viz Mosart can work with.

Common Settings

In most cases, the settings below are always required:

- **Initial Standby:** If selected, the server is forced to start in *Standby* mode.
- **Server Type:** Name of supported video server. For example, AirSpace, AirSpeed, EVS LinX, Orad OCIP.
- **Alias:** Defines the name of the server to display in the Viz Mosart GUI and Timing Display.
- **Name:** Internal name of the server, displayed in AV Automation

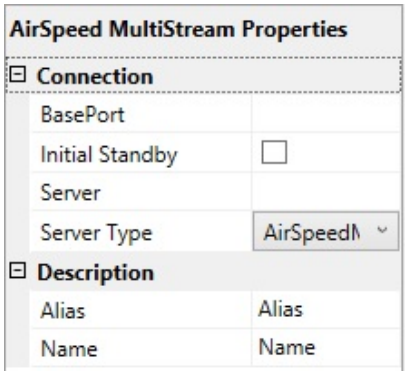
The above settings are typical for
AirSpace, AirSpeed, EVS LinX, EVS Xedio, OradOcip video servers.



Some common video server properties are summarized below:

- AirSpeed MultiStream
- AMP
- Grass Valley AMPP
- Grass Valley K2
- Advanced settings (not shown on UI)
- Nexio
- MVCP
- Omneon
- Quantel
- TriCaster
- VDCP
- VDCPtcp
- Viz Engine

AirSpeed MultiStream



- **BasePort:** The first port Viz Mosart uses to communicate with the AirSpeed MultiStream server. Default: `59451`
- **Server:** Defines the hostname or IP address of the video server.

AMP

Please refer to Common Settings above. In addition, there is

- **Port:** The default is 3811. Do not change this unless you know what you are doing.
- **Server:** Defines the hostname or IP address of the video server.

Grass Valley AMPP

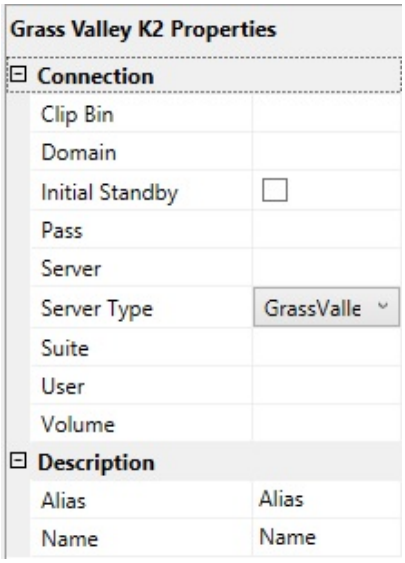
Please refer to Common Settings above. In addition, there is

- **Platform URL:** The AMPP Platform URL. This URL should look like *https://<region>.cloud.grassvalley.com*
- **API Key:** An API key allowing you to connect to the AMPP Platform for sending/receiving AMPP Control messages
- **Response timeout:** Timeout in milliseconds to wait for the responses from the server. Default 1000

Grass Valley K2

Connection to K2 is initialized with a ping (ICMP) from Mosart to the server to confirm the connection. Then it uses port 3811 to communicate with the K2 server. Both must be open for Mosart to be able to connect.
Media Administrator uses a separate port when it connects: 49171 (a range is OK. Use 49168-49172)

> **Note**: In some situations, cueing of a clip may fail due to an internal timeout in Viz Mosart. This can typically occur when trying to cue a clip that is *currently transfering* to the video server.
> See EventReceiverTimeout below.



- **Clip Bin**: Defines the Bin in a default server location for clips to be played. This default location is used only when the location is not provided by other means.
  Notably, it is *not* used when the setting **IgnoreBin** (in Media Administrator) is *true* for this server.
  The value of *Clip Bin* must designate a bin in the volume designated by the value of **Volume** (see below).
- **Domain**: The domain as part of the user credentials.
- **EventReceiverTimeout**: See below.
- **Pass:** The password as part of the user credentials.
- **Suite:** Arbitrary value. However, follow these guidelines:
  - For each K2 server, two or even three unique **Suite** values must be used:
    - First, the values for Media Administrator and AV Automation must be different.
    - Second, in a redundancy setup with Main and Backup Mosart server, the two values for Media Administrator must be different.

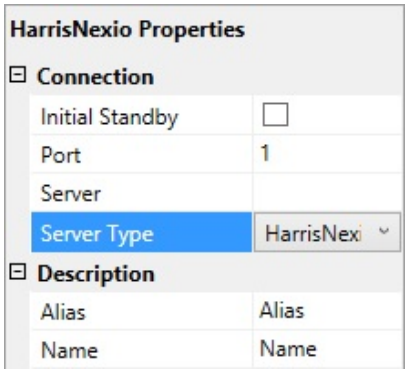
> **Info**: In this constellation, both Main and Backup AV Automation should use the *same* value, yet different from the two Media Administrator values.

- **User:** The username as part of the user credentials.
- **Volume:** This value, together with that of **Clip Bin** (see above) defines the default server volume location for clips that will be played. This default is only employed when a location has not been provided by other means. Notably, it is *not* used when the Media Administrator setting, **IgnoreBin** is *true* for this server.
- **Server:** Defines the hostname or IP address of the video server.

Advanced settings (not shown on UI)

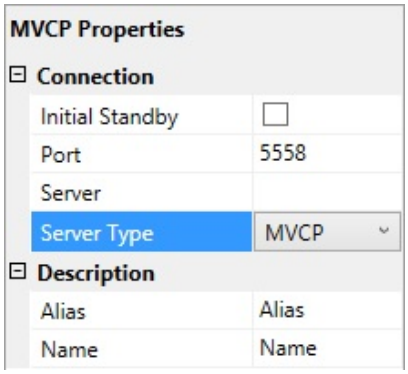
- **EventReceiverTimeout :** In some situations, cueing of a clip may fail due to an internal timeout in Viz Mosart. This can typically occur when trying to cue a clip that is *currently transfering* to the video server. In such cases one should increase the configurable timeout **EventReceiverTimeout** from the default *1000* msecs to, for example *4000* msecs. .
  - This advanced setting is found in AV Automation settings, opened with **Ctrl+Shift+S**.

Nexio



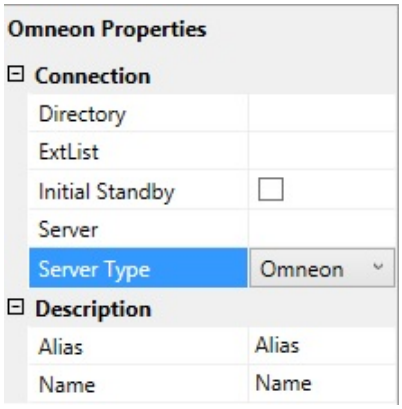
- **Port:** TCP/IP communication port to the Nexio server.
- **Server:** Defines the hostname or IP address of the video server.

MVCP



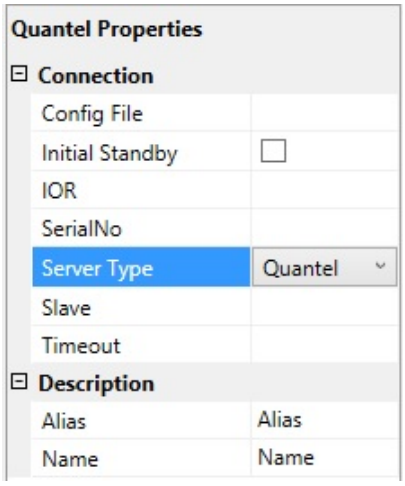
- **Port:** TCP/IP communication port to the MVCP server.
- **Server:** Defines the hostname or IP address of the video server.

Omneon



- **Directory:** Defines the directory where the clips are stored.
- **ExtList:** The list of valid file extensions used when listing and querying files on the server. The list is period separated and case sensitive.
- **Server:** Defines the hostname or IP address of the video server.

Quantel



- **Config File:** The path to the XML configuration file used to define the Viz Mosart - Quantel communication.
- **IOR:** The HTTP link including hostname or IP address to the IOR resource on the **Quantel** ISA manager. Example: http://192.168.100.50:2096/ZoneManager.ior
- **SerialNo:** The serial number of the Quantel playout server.
- **Slave:** The hostname or IP address of the slave/backup IOR.
- **Timeout:** Timeout value for requests from Viz Mosart to ISA manager. If the request exceeds this timeout the server connection is reinitialized. Setting a value here should only be needed for sites experiencing issues with the Quantel connections. Leaving a blank value uses the default timeout.

Copyright © 2026 Vizrt

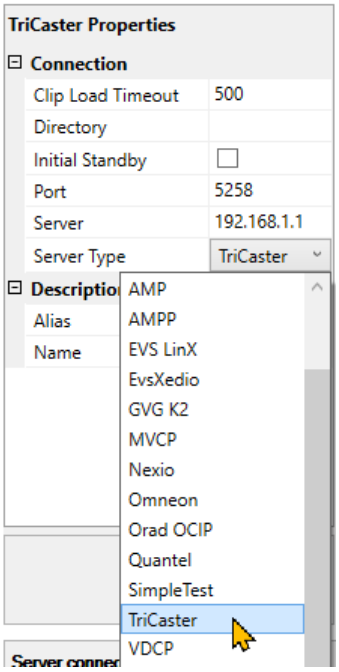
- **Server:** Defines the hostname or IP address of the video server.

Recording with Quantel Devices

Quantel requires a duration when preparing a recording. This duration can be set the following ways:

- As part of a template: Specified using *markin* and *markout* properties.
- As part of a keyboard shortcut: Specified by trailing the `ClipName` property with [,duration]. Similar to the examples under the general section Controlling Video Servers from Mosart above.
- As part of template control command: Specified as part of the **Parameter** property. Again, see the general section above.

TriCaster



The Vizrt TriCaster uses the TriCaster protocol (previously named *vizrt-ips* protocol).

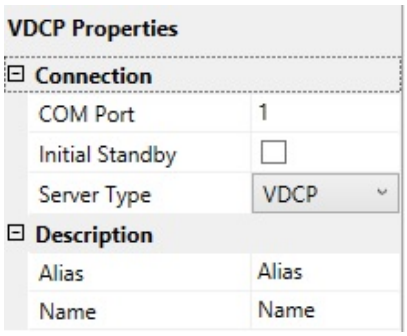
- **Clip Load Timeout:** Timeout in milliseconds to wait for a clip from a media server to be available. Default: *500 mSec*.
- **Directory**: The TriCaster directory used for clips path.
- **Initial Standby**: If selected, the server is forced to start in *Standby* mode.
- **Port**: TCP/IP communication port to the TriCaster (Default: 5951).
- **Server**: Defines the hostname or IP address of the TriCaster.
- **Server Type**: TriCaster.

All the settings for TriCaster can be stored in ClipServerTriCaster.xml (usually in `cd %ProgramData%\Mosart Medialab\ConfigurationFiles\`

To address usage settings from this file the connection string for the TriCaster video server can be simplified:

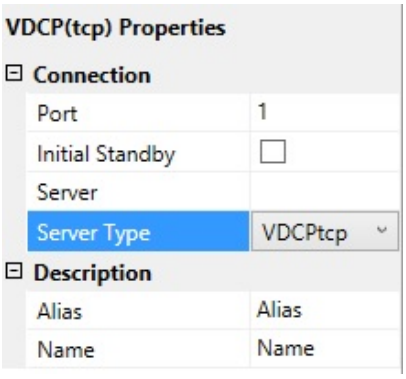
```
Name=TC-mosartvmc;Type=TriCaster;Config=ClipServerTriCaster.xml;
```

VDCP



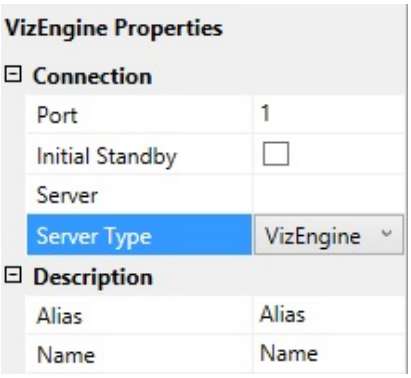
- **COM Port:** The COM port connected to the VDCP video server.

VDCPtcp



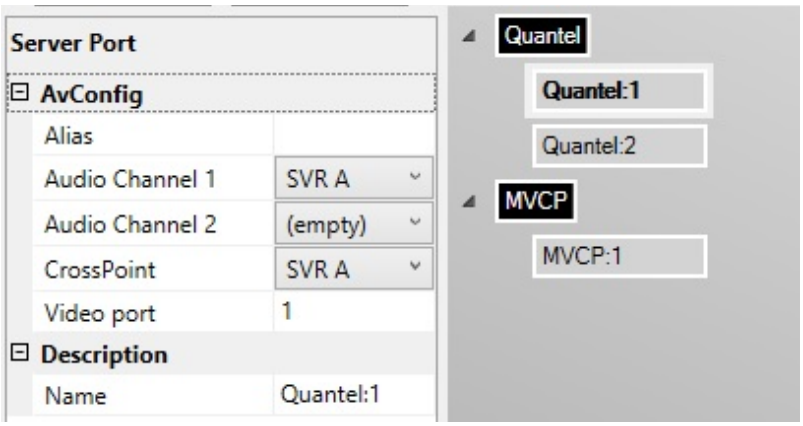
- **Port:** The TCP/IP communication port connected to the VDCP video server.
- **Server:** Defines the hostname or IP address of the video server.

Viz Engine



- **Port:** The TCP/IP communication port connected to the Viz Engine video server.
- **Server:** Defines the hostname or IP address of the video server.

Video Server Port Properties



- **Alias:** Defines the name of the server to display in the Viz Mosart GUI and Timing Display.
- **Audio Channel 1:** The audio channel 1 for the selected video port.
- **Audio Channel 2:** The audio channel 2 for the selected video port.
- **CrossPoint:** The video crosspoint for the selected video port.
- **Video port:** The actual port name/number on the video server. Please refer to the documentation of your video server for information on how to find these.

> **Info**: For the Server Type *TriCaster*, any recording port must have a 'Video port' of the form recN, where N is a number.

- **Name**

Working with Servers and Ports

To Add a Server

1. Click the **Add new server** button
2. In the **Properties** pane on the left, select from **Server Type** and add additional information such as **Port** and **Server host**.

To Remove a Server

- In the **Video Server Configuration** tree, right-click a server node and select **Remove**.

To Add a Port

1. In the **Video Server Configuration** tree, select a server node
2. Click the **Add new port** button, or right-click the server node and select **Add new port**
3. In the property editor on the left, enter the port information.

To Remove a Port

- In the **Video Server Configuration** tree, right-click a port node and select **Remove**.

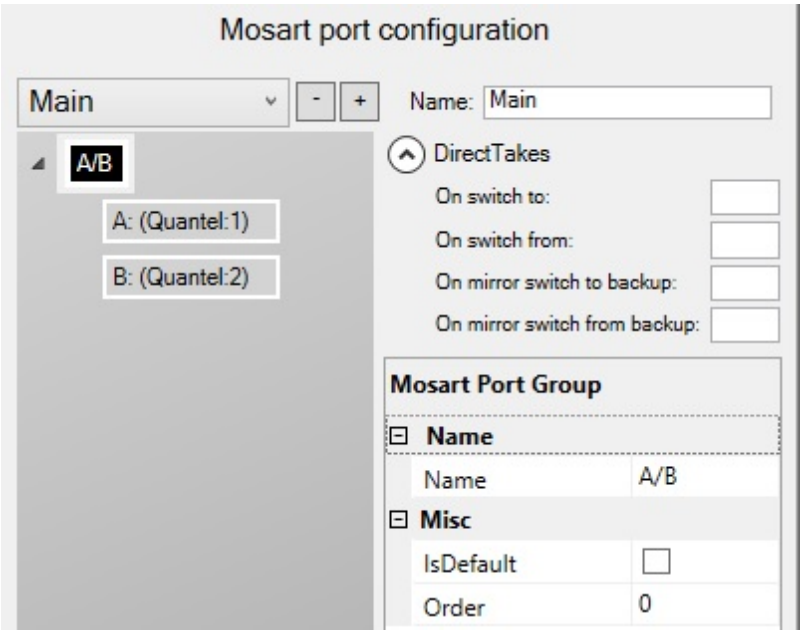
Mosart Port Configurations

The right-hand part of the Video Servers tab (titled 'Mosart port configuration', shown in the screenshot below) has three parts:

- the *salvo* part (in the screenshot the first ‘line', starting and ending with 'Main’, and also the DirectTakes panel with four text boxes and labels)
- a tree structure showing *virtual ports* and *virtual port groups* (the bulk of the left-hand side of the screenshot)
- a panel showing the *details* of the virtual port or the virtual port group selected ('Mosart Port Group' in the screenshot).

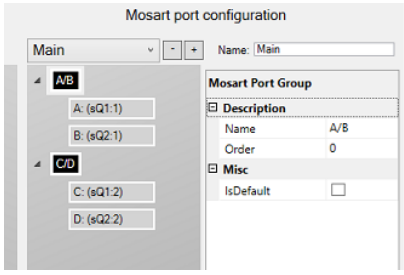
A salvo is a particular setup of servers and ports. There will always exist at least one salvo. If several salvos are defined, these will usually involve different sets of servers and/or ports, and the user will be able to *switch* from the *active* salvo to some other salvo. This will involve, among other things, *disconnecting* from servers that will not be used in the salvo switched to, and *connecting* to servers that are to be used now. This will be discussed later.

Copyright © 2026 Vizrt



Working with Mosart Port Configurations

The tree structure shows the *virtual ports* and *virtual port groups* defined. (These are the same in all salvos.) Visually (in the tree structure), the groups are black, and the ports are grey (and indented). A virtual port may have one or two server ports linked to it. (If it has two server ports linked to it, the two are *mirrored*. Mirroring will be discussed later.) A virtual port group is, as the name indicates, a group of virtual ports. Such a group may be used for A/B roll (playout of clips on alternate ports). Both virtual port groups and individual virtual ports are available for selection in the Template editor.



To add a virtual port group

- Right-click in the **Mosart Port Configuration** tree (not on a node) and select either:
  - **Add virtual ripple group** to add a new virtual ripple group. The default ripple group names are *A/B, C/D, E/F* etc.
  - **Add virtual preview group** to add a new green virtual preview group to be used for preview ports. The default preview group name is *P*.
  - **Add virtual recording group** to add a new red virtual recording group to be used for recording ports. The default recording group name is *Rec* .

> **Info**: The virtual port group is added to *all* salvos.

You can edit or rename the selected group using the **Properties Editor** on the right.

To remove a virtual port group

You can remove a virtual ripple group, preview group or recording group.

1. In the **Mosart Port Configuration** tree, right-click any ripple/preview/recording group and select **Remove group <name>**.
   Alternatively, select a group and press the **Delete** key.
2. When prompted, click *Yes*. The group will then be removed from all salvos. (If you click *No*, nothing will happen.)

To add a virtual port

- In the **Mosart Port Configuration** tree, right-click any group and select **Add virtual port**.

If the port was added to a *preview* group, the added port will be a preview port with default name *P*. If the port was added to a *recording* group, the added port will be a recording port with default name *Rec*.

> **Note:** Do not change the name *Rec*!

If the port was added to a virtual *ripple* group, the added port will be a playout port. The name will be the first unused (for this purpose) (upper-case) Latin letter (A, …, Z).

> Info: There is no relationship between the port name and the group name. For example, group A/B may have ports C and D, and vice versa.

To remove a virtual port

1. In the **Mosart Port Configuration** tree, right-click any *virtual port* and select **Remove <name>**. Alternatively, select a port and press the **Delete** key.
2. When prompted, click *Yes*. The port will then be removed from all salvos. (If you click *No*, nothing will happen.)

To link a server port to a virtual port

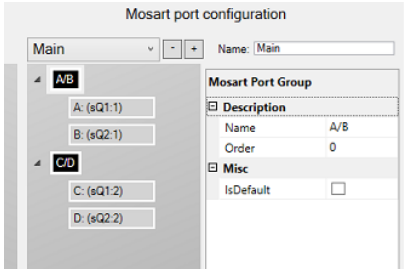
You can link a server port to a virtual port:

- Select the server port (in the left-hand, Video server configuration part) you want to link, and drag it to the virtual port (in the right-hand Mosart port configuration part) to which you want to link it.

> **Note:** If the virtual port is a *recording* port (in the recording group), the server port to link must also be a recording port: The physical server must have recording capabilities (and the chosen Server Type must allow for recording).

To inspect or change the details of a virtual port group

Select a virtual port group in the tree structure:

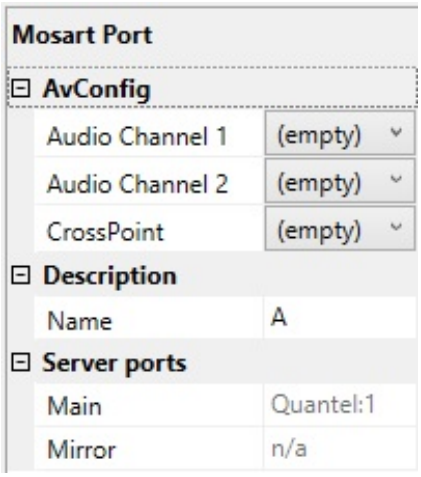


(Here, A/B is selected.) These details are available:

- **Name:** The name of the virtual port group. This name is displayed in the will be among the options in the SERVERCHANNEL drop-down in the Videoserver Device properties in the Template editor.
- **IsDefault:** Enable to set the default and preferred video port group for assets residing on multiple systems.
- **Order:** Used by the Media Router only. (See the documentation of the Media Router.) If a Media Router is not used, this value can be left at the default *0*.

To inspect or change the details of a virtual port

Select a virtual port in the tree structure:



(Here, A is selected.) These details are available:

- **Audio Channel 1:** The virtual audio fader 1 for the selected video port. If nothing is selected here, the audio fader from the physical port is used. Default: *Empty*
- **Audio Channel 2:** The virtual audio fader 2 for the selected video port. If nothing is selected here, the audio fader from the physical port is used. Default: *Empty*
- **CrossPoint:** The virtual video crosspoint for the selected video port. If nothing is selected here, the video crosspoint from the physical port is used. Default: *Empty*

Note: In most cases, the three fields above should be left at their default '(empty)'. Then the fader/crosspoint from the physical port will be used.
Changing to a specific fader/crosspoint will override this. Do this only if you know what you are doing.

- **Name:** The name of the virtual port. This name will be among the options in the SERVERCHANNEL drop-down in the Videoserver Device properties in the Template editor.
- **Main:** Auto-generated display name of the linked server port (as explained in To link a server port to a virtual port above), if any.
- **Mirror:** Auto-generated display name of the mirror server port. (This refers to mirroring, to be explained shortly.)

Mirroring

Mirroring refers to having two (instead of one) physical ports linked to each virtual port. (*Only two* ports may be mirrored.) The two ports will have the *same state*. E.g., they will both have the same clip cued, or they will both play the same clip (and at the same position). The intended use of this feature is to be a redundancy setup. At start-up, the ‘Main' ports is active, i.e., its outputs (video and audio) are routed through the mixers, whereas the outputs of the 'Mirror' ports are not. The Mosart operator may, especially if there is a problem with a video server linked to a Main port, *switch* to the 'Mirror’ ports. (There are functions for this both in the Viz Mosart GUI and in AV Automation.) Then the role of the ‘Main' and the 'Mirror' ports are *swapped*: Now the outputs of the *Mirror* ports are routed through the mixers, whereas the outputs of the Main ports are not. (The problem with the video server should then be fixed, so that the operator may switch back to Main at some point.)

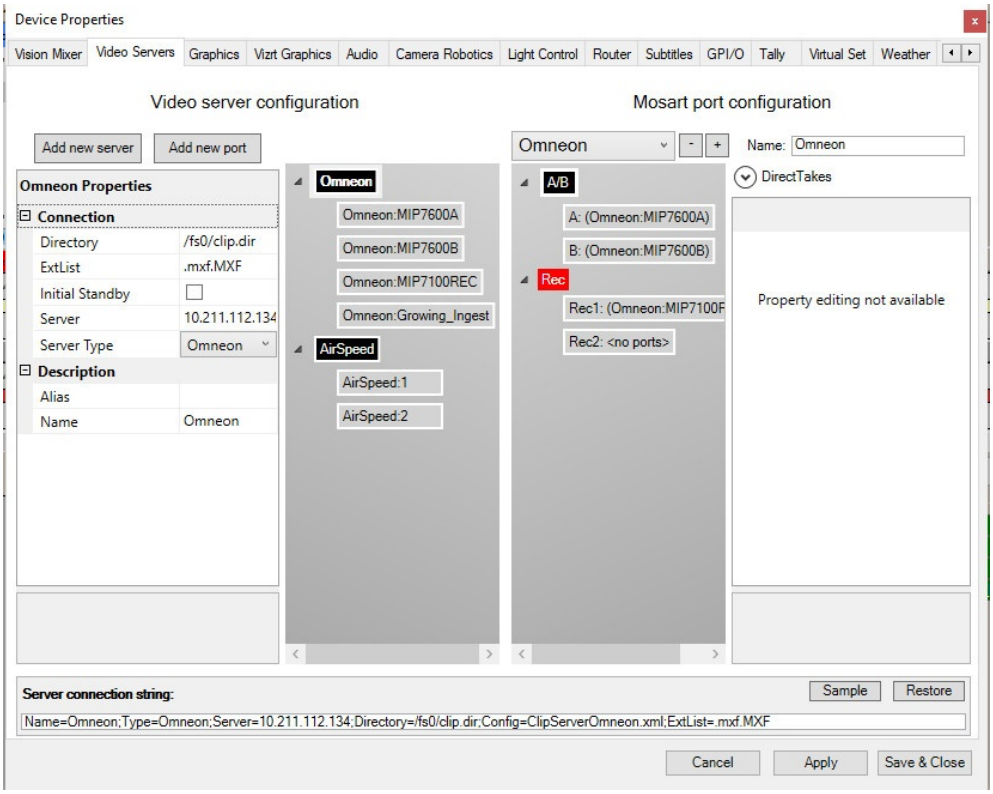
For this to work as intended, no physical server should participate in both Main and Mirror. I.e., if a physical server has at least one port being the Main port for some virtual port, it should not have a port being the Mirror port for some virtual port.

This method differs from a the use of Main and Backup salvos, to be described later.

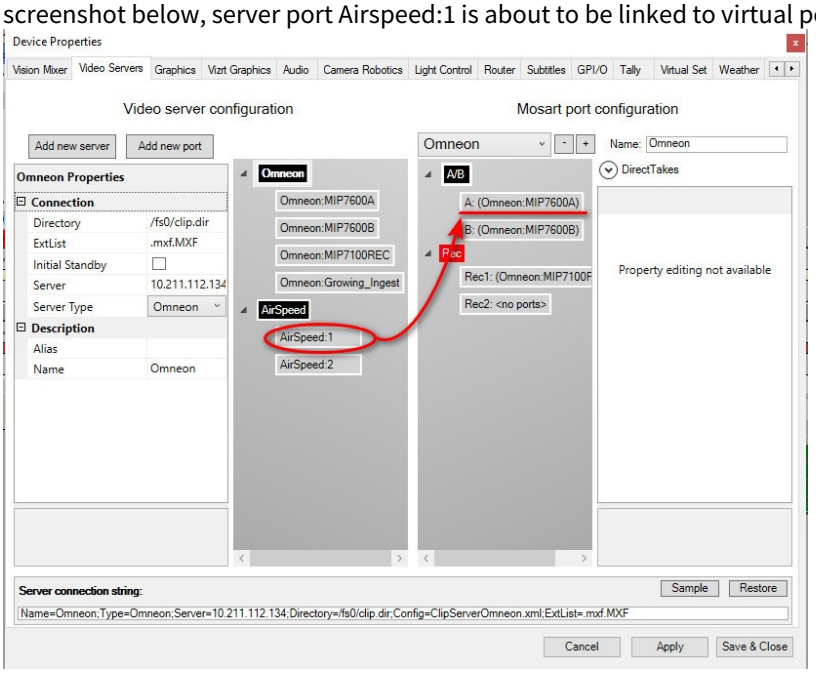
To link a second server port to a virtual port for mirroring

It is assumed that you have at least one physical video server and at least two physical ports.

1. Link one server port to a virtual port as described in To link a server port to a virtual port. Then the situation may be as in the screenshot below. (Here server port *Omneon:MIP7600A* has been linked to virtual port A.)

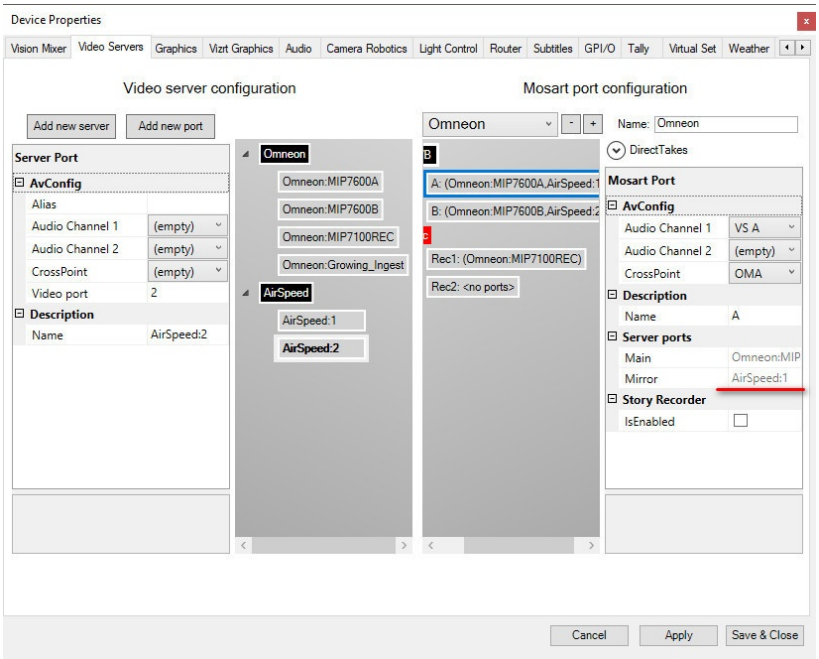


2. Link *another* server port to the *same* virtual port. (Referring to the general description of the Mirroring feature above, the two server ports linked to a virtual port should reside on different servers.) In the screenshot below, server port Airspeed:1 is about to be linked to virtual port A.

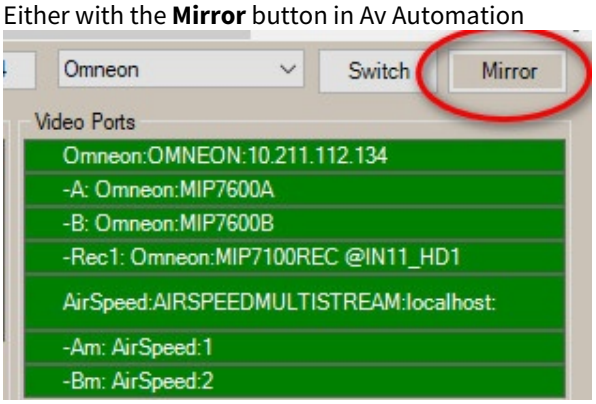


The text in the virtual port node in the tree will change to reflect that now *two* physical ports have been linked to it..

3. Click on the virtual port to verify that the server port from step **2** is now listed as a mirror. In the screenshot below, server port Airspeed:1 is indeed the Mirror linked to virtual port A.

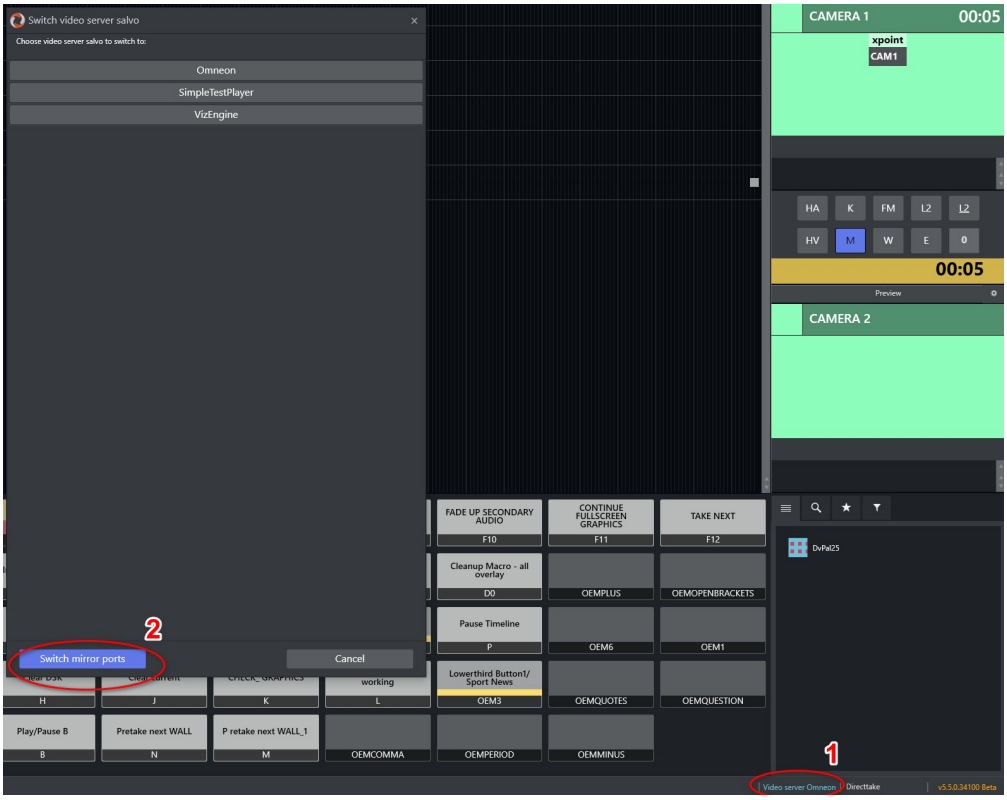


4. Repeat for other virtual ports as needed.
5. Click **Apply**, then **Save & Close**.
6. Verify that you can now switch between mirror ports
    a. Either with the **Mirror** button in Av Automation



or

    b. In the Mosart GUI.



To clear server links in a virtual port

- Right-click the virtual port and select **Clear links**.

To clear the server links from a virtual port group

- Right-click the virtual port group and select **Clear all links**.

Working with Salvos

A salvo is a particular setup of servers and ports. All salvos have the *exact same virtual ports and port groups*. Salvos differ only in the assignments of physical ports to virtual ports. There will always exist at least *one* salvo. If several salvos are defined, these will usually involve different sets of servers and/or ports. At any time, one salvo will be *active*. The user will be able to *switch* from the active salvo to some other salvo (which will then become active). This will involve, among other things, *disconnecting* from servers that will not be used in the salvo switched to, and *connecting* to servers that are to be used now.

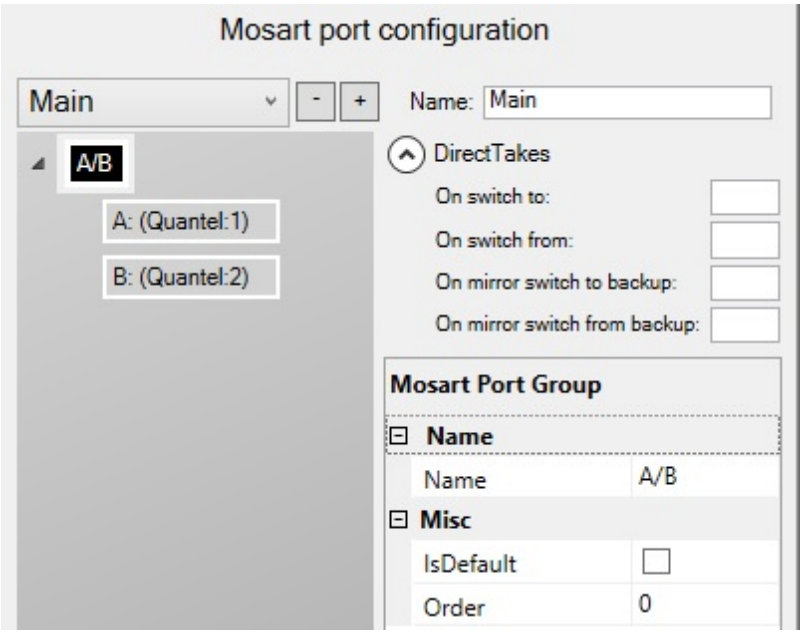
> **Info:** Salvos are distinct from Mirroring. When using Mirroring, both the Main and the Mirror ports are connected all the time.

There are several use cases for salvos:

- Redundancy: To lessen the effect of video server malfunctions, Main and Backup salvos may be defined, presumably using different servers.
- Different needs for different shows, or even within the same show (for instance, sports and news).

Info: It is recommended that the default *Main* and *Backup* salvos are not renamed.

The configuration of salvos are done in the upper line of ‘Mosart port configuration', and also in the 'DirectTakes’ panel:



The available UI elements are

- Drop-down with the configured salvos, used for selecting a salvo for configuration or deletion. (If another salvo is selected, this salvo will become active. However, this is not the standard way of switching salvos during normal operation. The standard method is using the salvo controls in the main window in AV Automation.)
- button for deleting the selected salvo
- button for adding a salvo
- **Name:** The name of the selected salvo.
- **Direct Takes**
  - **On switch to:** The Recall Nr (see Template Properties) of the direct take template that should be taken when switching to this video server salvo.
  - **On switch from:** The Recall Nr of the direct take template that should be taken when switching from this video server salvo.
  - **On mirror switch to backup:** The Recall Nr of the direct take template that should be taken when switching to a backup server using this video server salvo.
  - **On mirror switch from backup:** The Recall Nr of the direct take template that should be taken when switching from a backup server using this video server salvo.

To add a salvo

1. Click the button.

A new salvo is created and automatically selected in the drop-down (and also becomes the active salvo). Please note that there are no linking of physical ports to virtual ports. You may want to make such links, and also to change the name of the new salvo.

To remove a salvo

1. Select the salvo to remove in the drop-down.
2. Click the - button.

The selected salvo is removed. Another salvo is automatically selected in the drop-down (and also becomes the active salvo).

To modify a salvo

1. Click the drop down box above the **Mosart Port Configuration** tree and select the salvo you want to edit.
2. Rename the selected salvo in the **Name** textbox *and/or*
3. Add/change port links as described above.

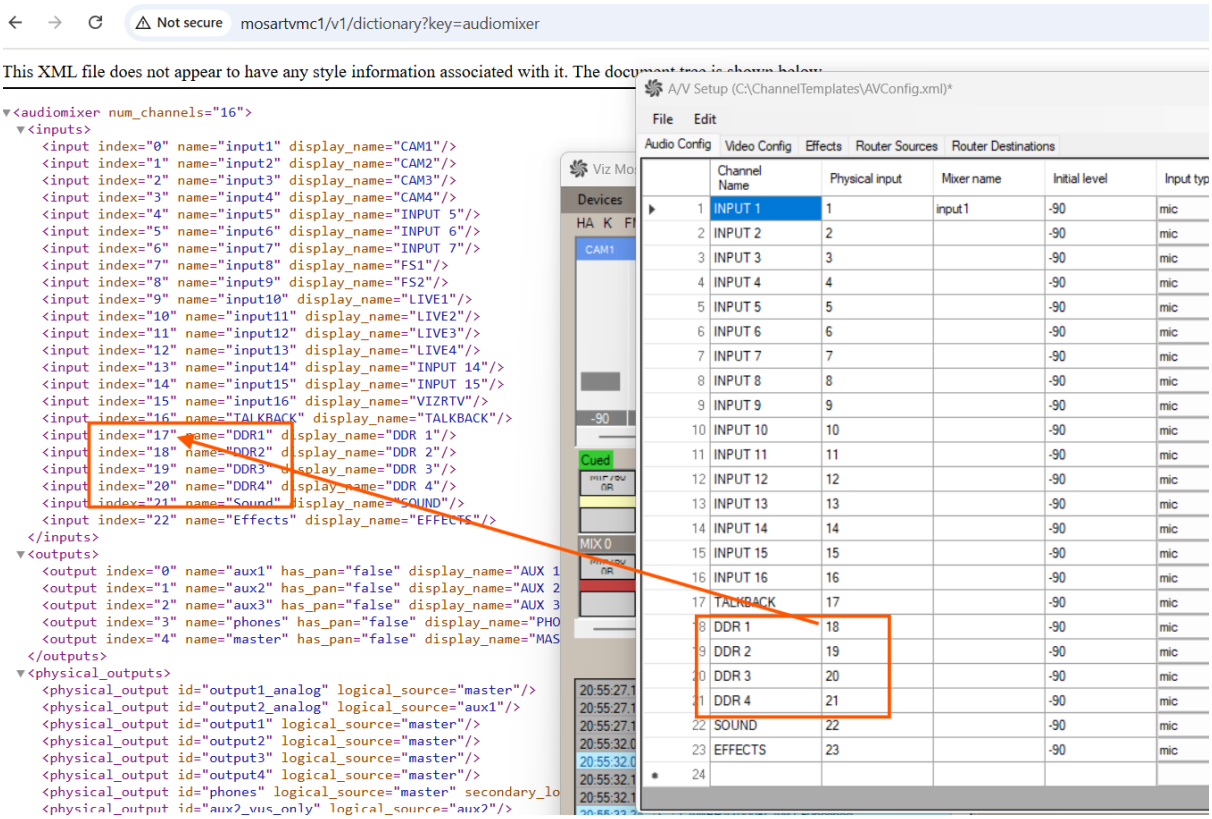
Video Server TriCaster Setup Example

As an example of a typical video server workflow, this scenario describes how Viz Mosart can control the clip playout and the recording of the output from a TriCaster video mixer.

Configure the TriCaster audio and video channels in Mosart

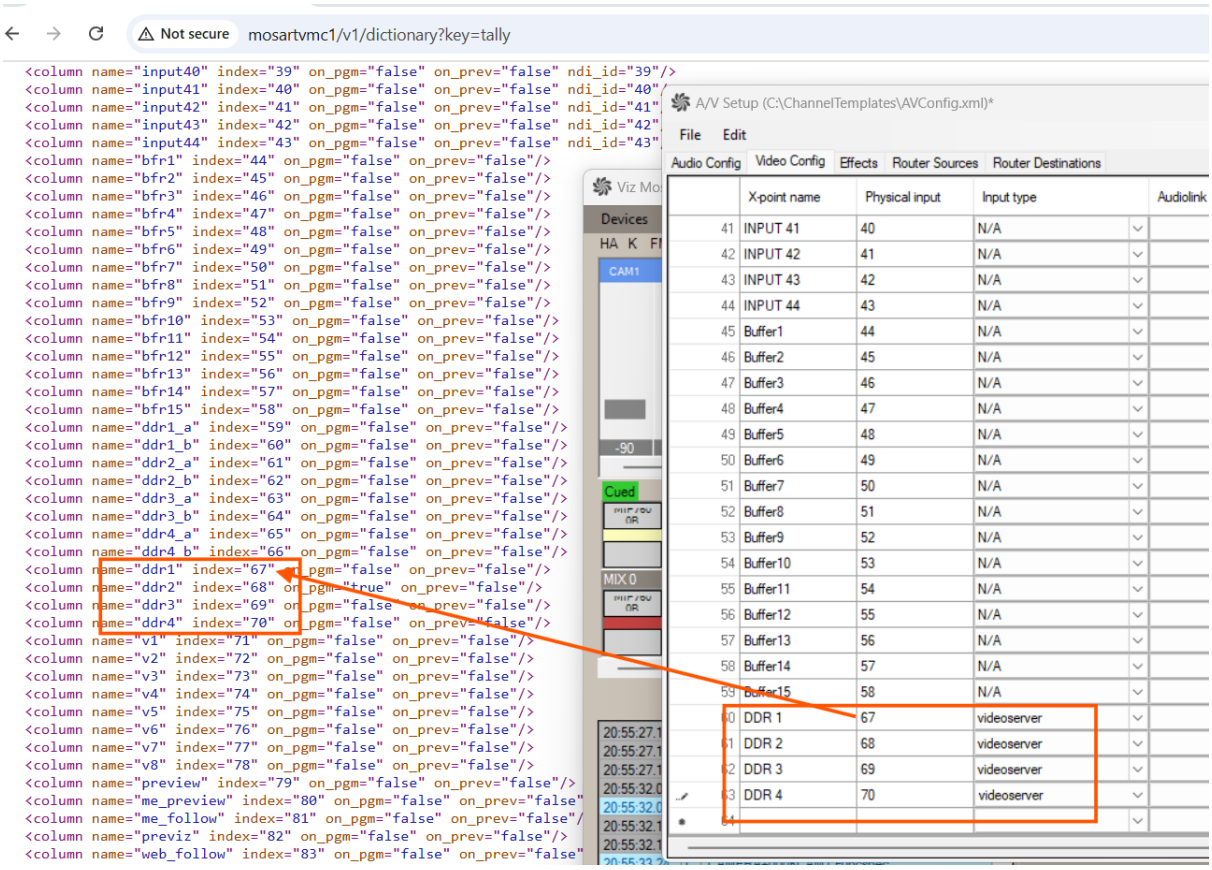
Before configuring **TriCaster as a video server** in **Viz Mosart**, ensure that the **audio and video inputs** are correctly mapped in **Viz Mosart's A/V Setup**.

1. Navigate to **Av Automation > Devices > AV Setup > Audio config** and configure the audio inputs as configured on TriCaster.
   The input indexes can be taken by accessing TriCaster over REST API:
   http://<TriCaster_host>/v1/dictionary?key=audiomixer
   For audio, the indexes are 1-based, so add +1 to the number found in the the request response:

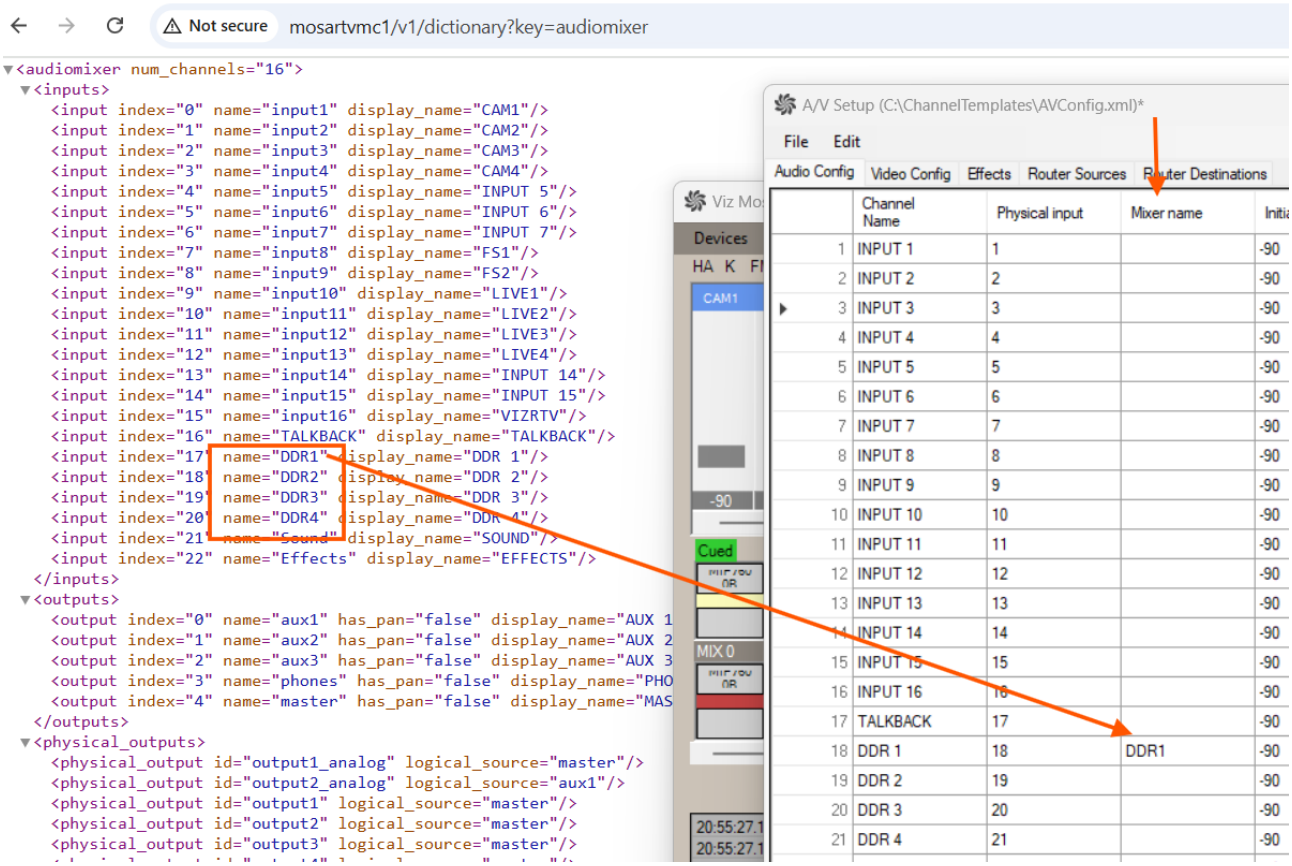


Alternatively, the input name can be mapped to the *Mixer name* column in A/V Setup in which case the Physical input value is not relevant, but must be filled to a number unique within the values from that column.

2. Navigate to **AV Automation > Devices > AV Setup > Video config**.
3. Add the crosspoint mapping video server ports as configured on the TriCaster.

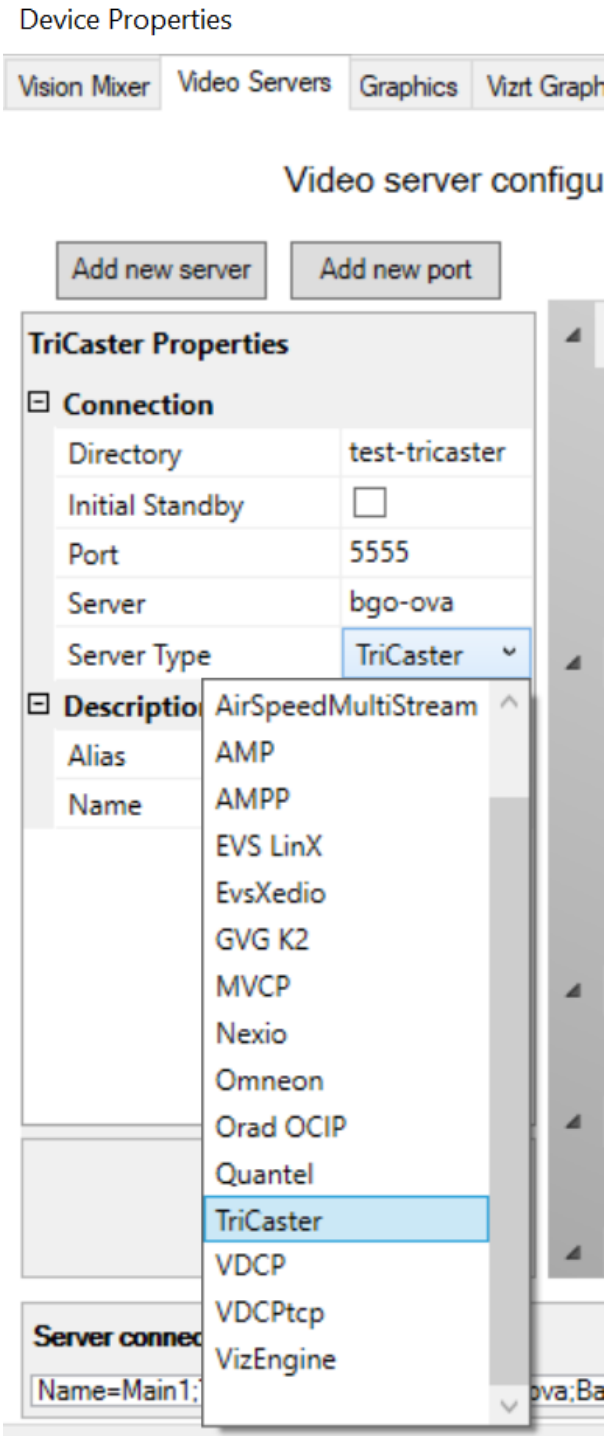


The input indexes can be taken by accessing TriCaster over REST API:
http://<TriCaster_host>/v1/dictionary?key=tally

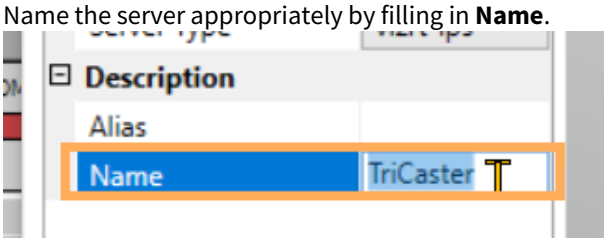


Configure TriCaster as video server

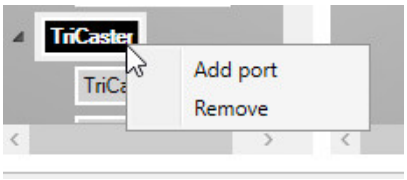
1. Navigate to **AV Automation > Devices > Properties > Video Servers**.
2. Under **Video server configuration** on the left side, click **Add new server**
3. If not in focus, click on the newly created black box, *New server*.
4. From the **Server Type** drop-down menu, select *TriCaster*.



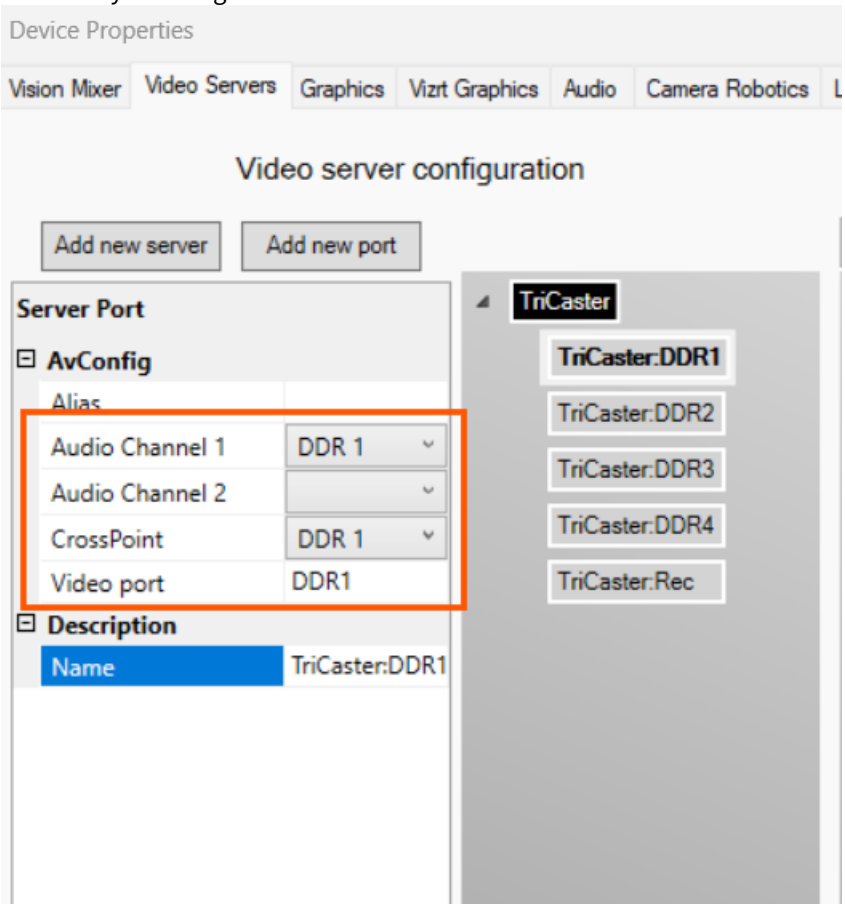
5. Name the server appropriately by filling in **Name**.



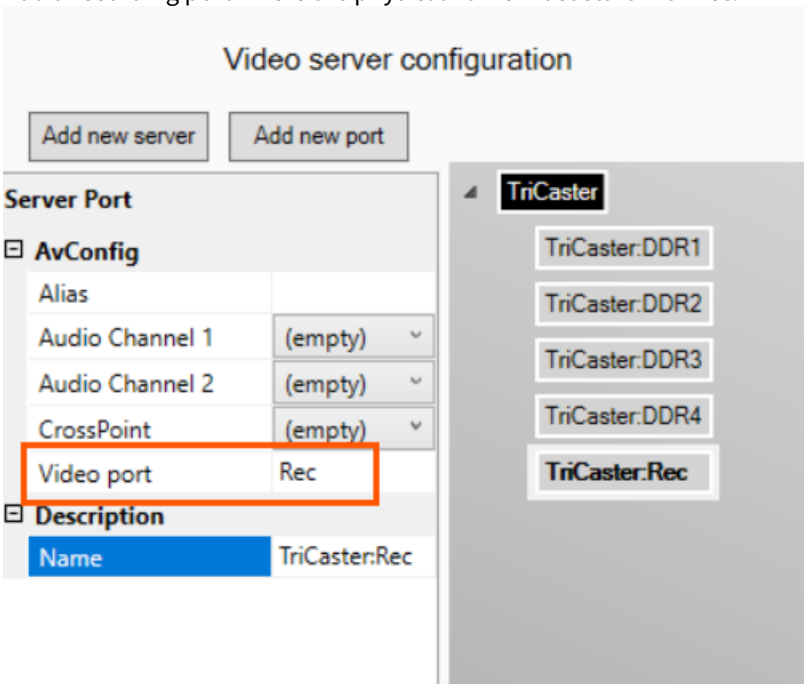
6. Click on the black box (see below) and configure the following properties for the server:
   Server: http://<TriCaster_host> as IP number.
   Directory: {volume}:/Sessions/{session name}/Clips/Import
   See *To find session name* below.
   Extension: let it empty (when Media Service is used, no extension is needed).
   Note: The clip id can be given from NRCS without extension.
   Name: Any meaningful name. Make sure the name is the same as the connection string provided in Media Admin.
7. Right-click the black box and add a port corresponding to each DDR port configured in TriCaster, DDR1, DDR2 etc.



8. For each server port added on the left side, configure:
   Audio channel 1: Select the audio channel configured in *A/V Setup > Audio Config* that is associated with the first mono channel of the stereo pair for playback from a DDR port in TriCaster.
   Audio channel 2: If applicable, select the audio channel in *A/V Setup > Audio Config* that is associated with the second mono channel of the stereo pair for playback from a DDR port in TriCaster.
   CrossPoint: Select the video channel configured in *A/V Setup > Video Config* that is associated with a playout ports in TriCaster.
   Video port: The physical port names as configured in TriCaster, DDR1, DDR2, DDR3 etc.
   Name: Any meaningful name.



9. Add a recording port where the physical name must start with *Rec:*

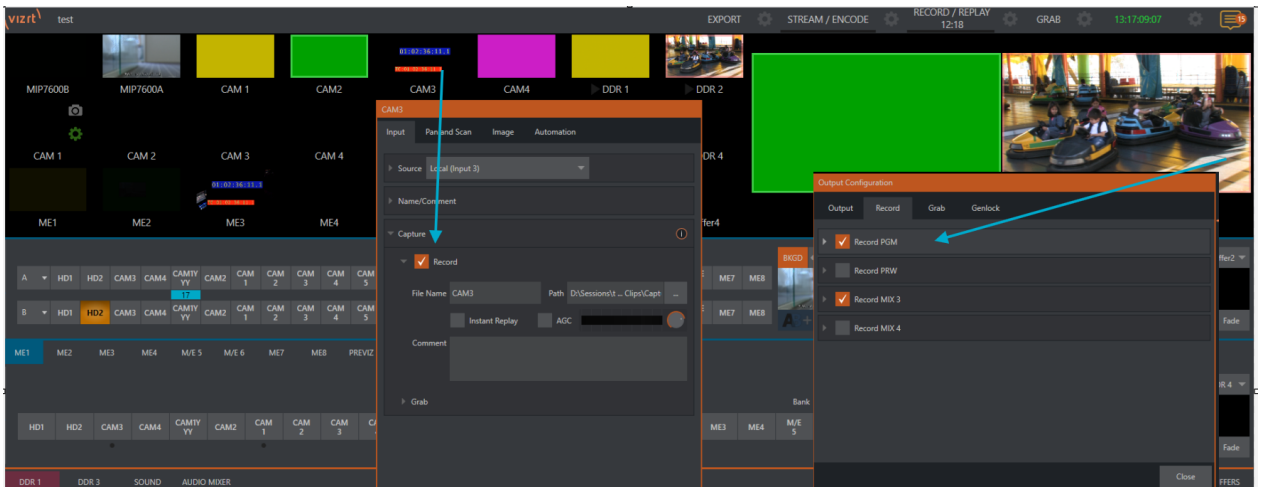


Info: From Viz Mosart, you can start the global TriCaster recorder, triggering the recording of any configured recording sources.
TriCaster offers two types of recording options:

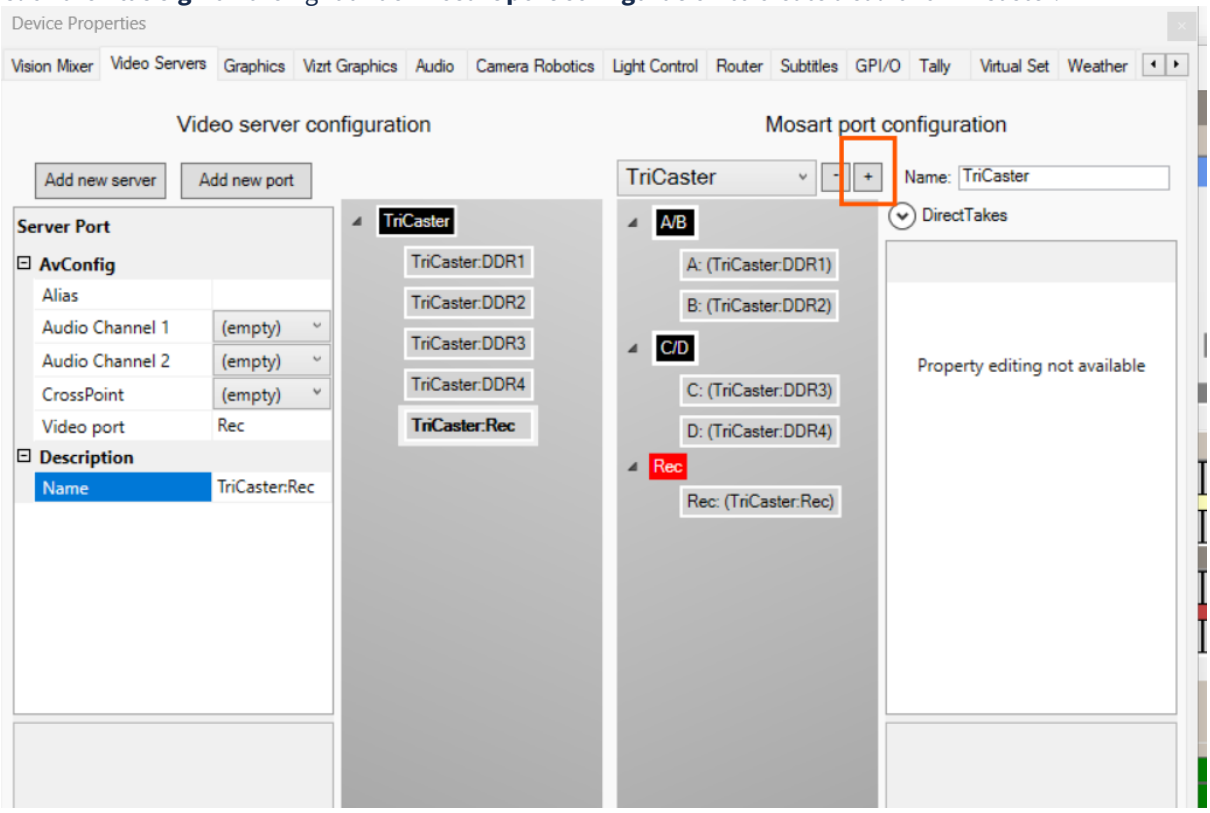
- MIX Output Recording: Records one of the four MIX outputs (PGM or other selected outputs).
- ISORecorder: Records specific input sources configured for isolated recording.

When recording is triggered from Mosart, any MIX output or ISORecorder inputs that have been enabled for recording in TriCaster will automatically start recording.
The recorded files are stored based on the recording configuration set in TriCaster's record settings.

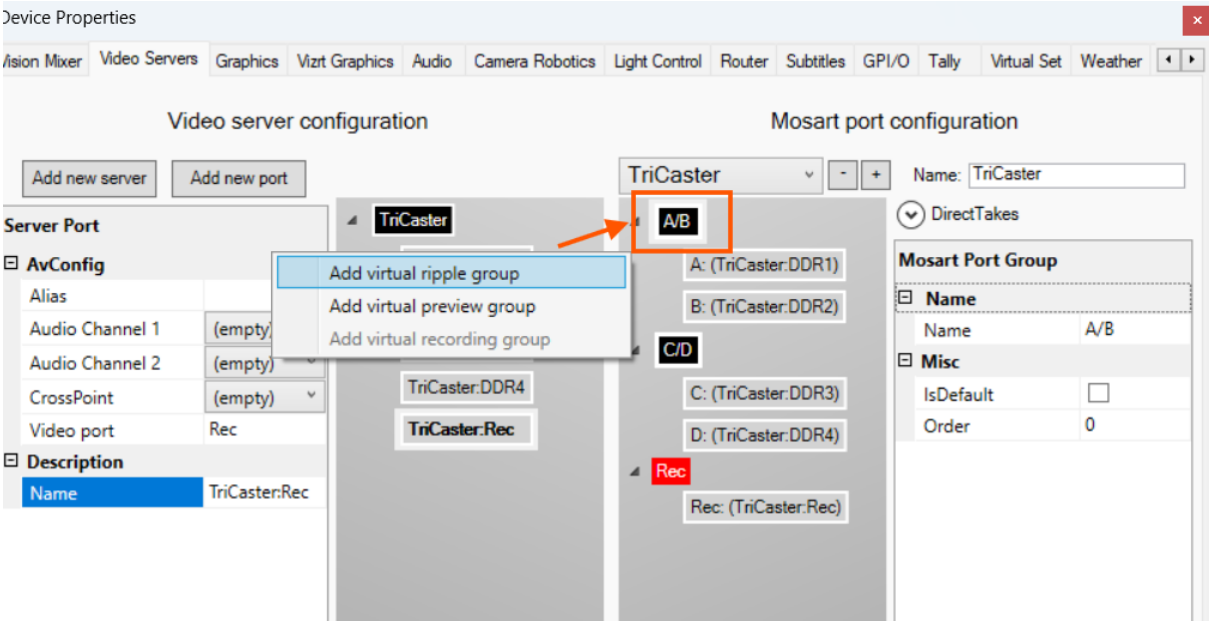
Note: *Ensure that the necessary sources are* ***enabled*** *for recording in* ***TriCaster's settings*** *before triggering the recorder from Mosart.*



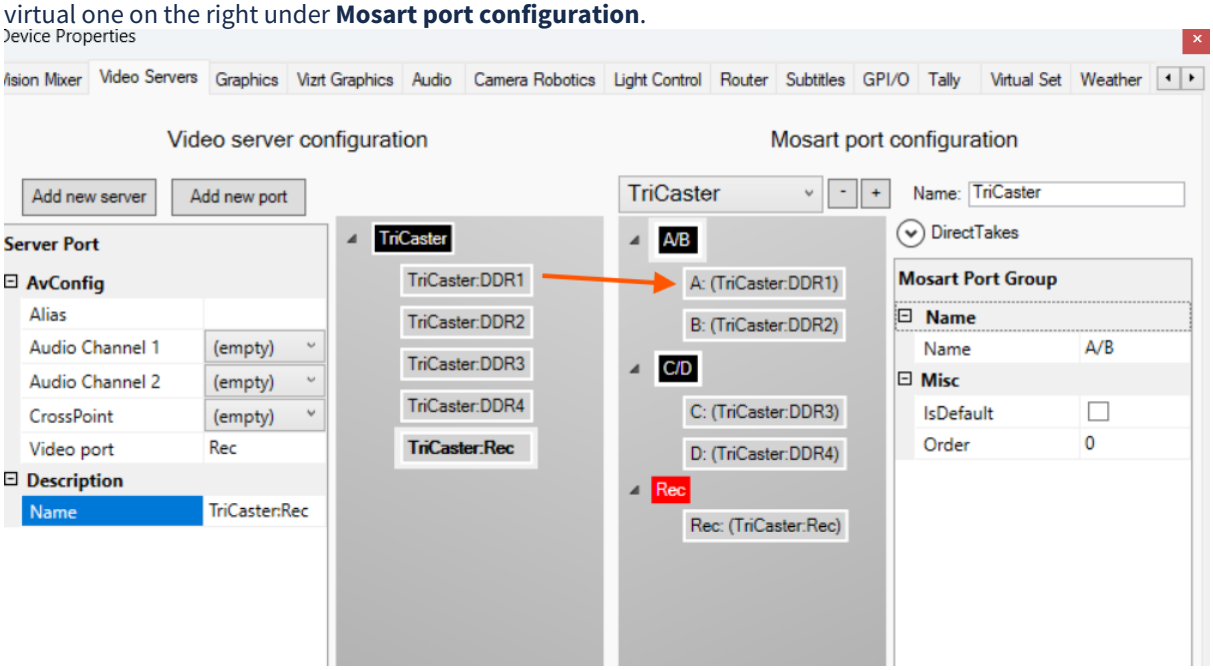
9. Click the **Plus sign** on the right under **Mosart port configuration** to create a salvo for TriCaster:



10. In **Name**, give this new virtual server an appropriate identity, for example, *TriCaster*.
11. Right-click on the *black* box to add a virtual port group.



12. Click on the first **Video server configuration** (physical) port on the left half and drag-and-drop it to the first virtual one on the right under **Mosart port configuration**.

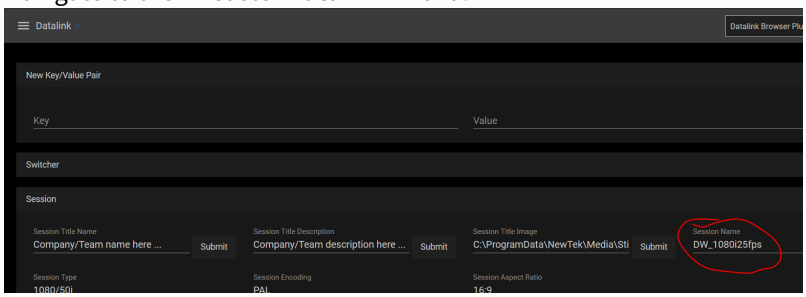


13. Repeat with the second port, by dragging another physical server port from the left side to pair them into an A/B ripple port playout group.
14. Add another virtual ripple group with 2 virtual ports and link them with the rest of the physical ports, if any.
15. A virtual recording group with one virtual port and link the physical recorder port to the virtual one in the same way as explained in step 12.
16. Click **Apply**, then **Save & Close**.
17. Restart AV Automation if needed.

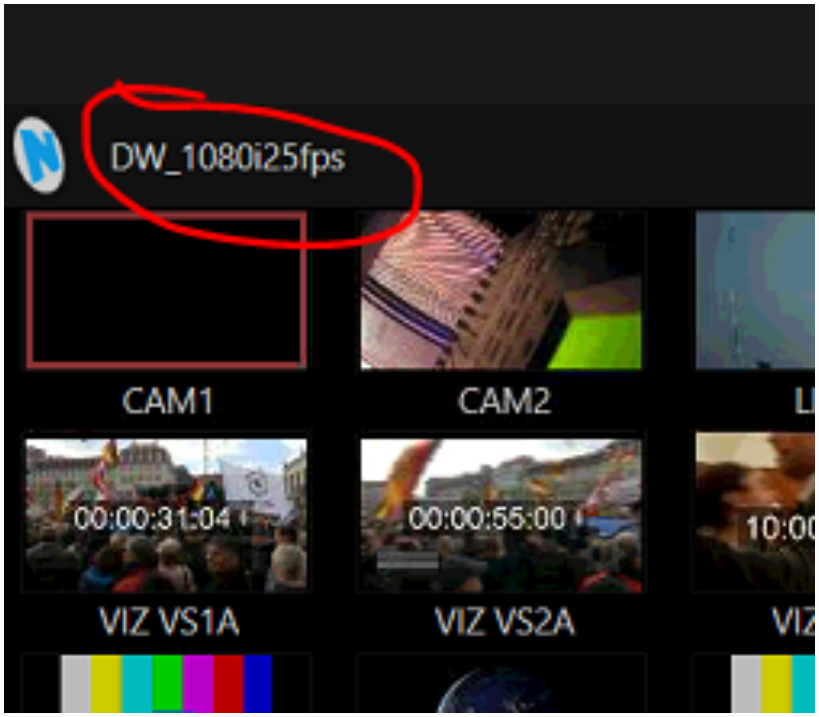
To find session name

Info: To find *session name*:

- Navigate to the TriCaster **DataLink** menu:



- The session name is also displayed on the TriCaster home page (hidden menu at the top left corner):


DW_1080i25fps
CAM1
CAM2
00:00:31:04
00:00:55:00
VIZ VS1A
VIZ VS2A

5.4.3 Graphics

Viz Mosart is capable of connecting to a variety of graphic engines available from many different manufacturers.

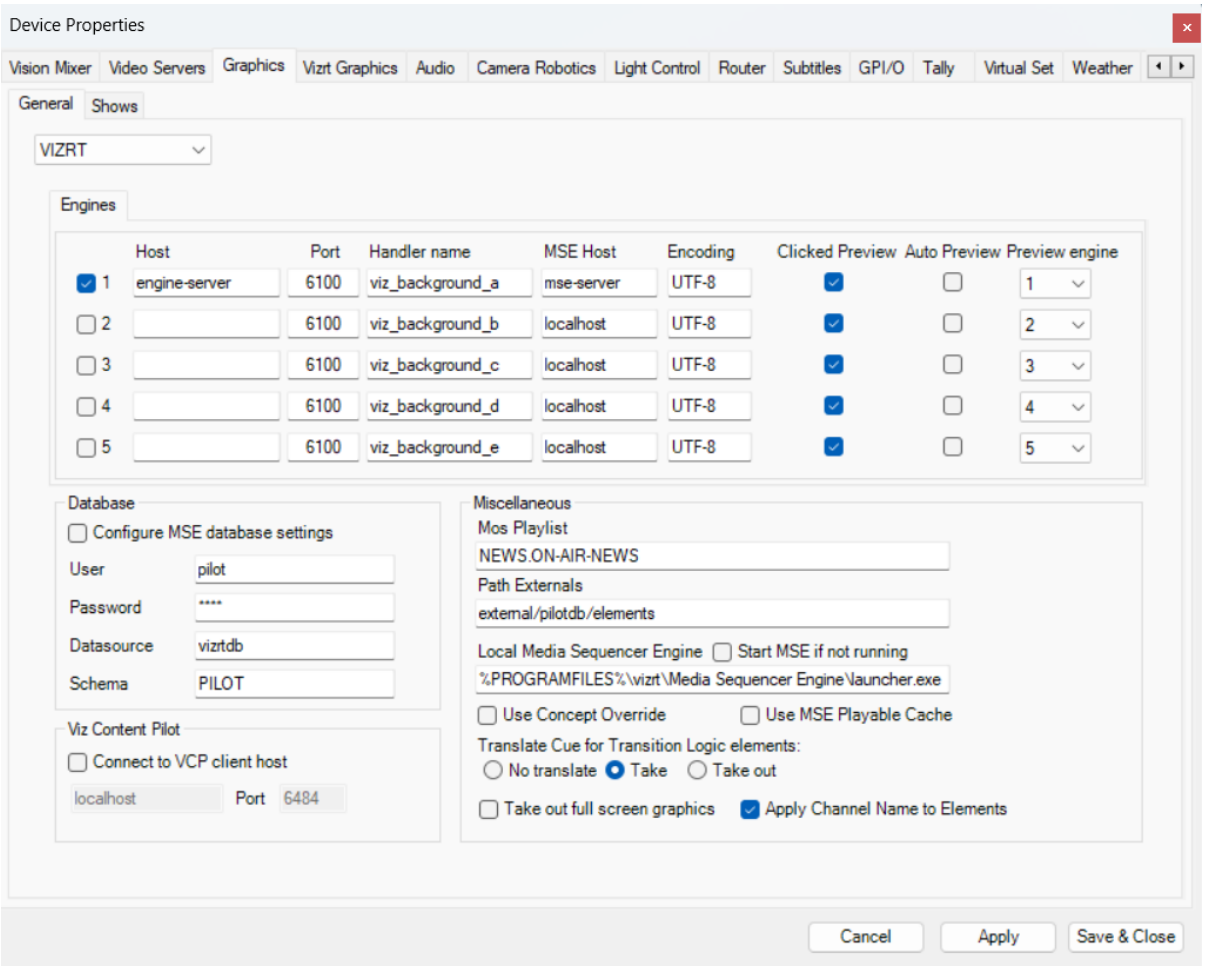
AV Automation handles full frame graphics playout, from a primary template type in the rundown.

The properties available depend on the system to be used:

- Vizrt Settings
  - General Tab
    - Engines
    - Database
    - Miscellaneous
    - Advanced settings (not shown on UI)
  - Shows Tab
- Viz Flowics
- Deko Settings
- XPression Settings
- Orad Settings
- Pixel Power Settings
- Chyron Settings

Copyright © 2026 Vizrt

Vizrt Settings



General Tab

Engines

Enables connection to each Viz Engine. Viz Mosart supports five graphics engine connections when checked.

- **Host:** Hostname or IP address to Viz Engine.

> **Note:** A port other than the default (6100) should be added as an appendix to the host. For example: 127.0.0.1**:6101** and *not* to the **Port** field.

Mirroring
You can add multiple Vizrt Viz Engines to any of the five graphics connections.

- Separate the engine name with a **comma** (**,**).
For example: *engine1:port1, engine2:port2, engine3:port3*

> **Info**: All mirrored engines will share the same MSE.
> - You can add up to 9 engines.

- **Port:** PDB only: Port to communicate with the Viz Engine. Default: *6100*.
  - **Handler name:** Media Sequencer only: Internal Media Sequencer handler name.
  - **MSE Host:** Media Sequencer only: Hostname or IP address to the machine running the Media Sequencer.

    > **Note:** For Media Sequencer, *both* **Handler name** and **MSE Host** must be specified.

    > **Info**: Creating a backup MSE: You can define a backup MSE here by adding a second IP after a "**;**"

  - **Encoding:** Font encoding for the Viz Engine. Default: *UTF-8*.
  - **Clicked Preview:** Enables the “click on full screen graphic element” to be taken on the preview engine.
  - **Auto Preview:** Enable to automatically take all full screen graphic elements in the preview engine.
  - **Preview engine:** The number of the Viz Engine that is selected to be the preview engine.

Database

- **Configure MSE database settings:** Enable to update the database settings in the Media Sequencer with the details below.
  - **User:** Username on the Oracle database. Default: See section *Security*.
  - **Password:** Password on the Oracle database. Default: See section *Security*.
  - **Datasource:** TNS name or connection string of the Oracle database. Default: See section *Security*..
  - **Schema:** Oracle database schema for the Viz Pilot connection. Default: See section *Security*..
- **Viz Content Pilot**
  - **Connect to VCP client host** (check-box):
    When checked, enables playout of Viz Pilot elements directly through a Viz Pilot client.
    Use of Viz Pilot must be assigned for each template, and the special Viz Pilot macros `mosart_load`, `mosart_start` and `mosart_continue` must exist in Viz Pilot.

    > **Note:** The "Display name" of both configurations must be exactly the same

  - **Host:** Hostname or IP address to the Viz Pilot client.
  - **Port:** Port to connect to the Viz Pilot macro port.

Miscellaneous

- **Mos Playlist:** Playlist in VCP that is populated through the Vizrt MOS gateway. Default: *NEWS.ON-AIR-NEWS*

  > **Info:** The playlist name used here cannot be the same as that used in **Trio Interface > Engines Setup** Tab **> Playlist >** [*Playlist name*].

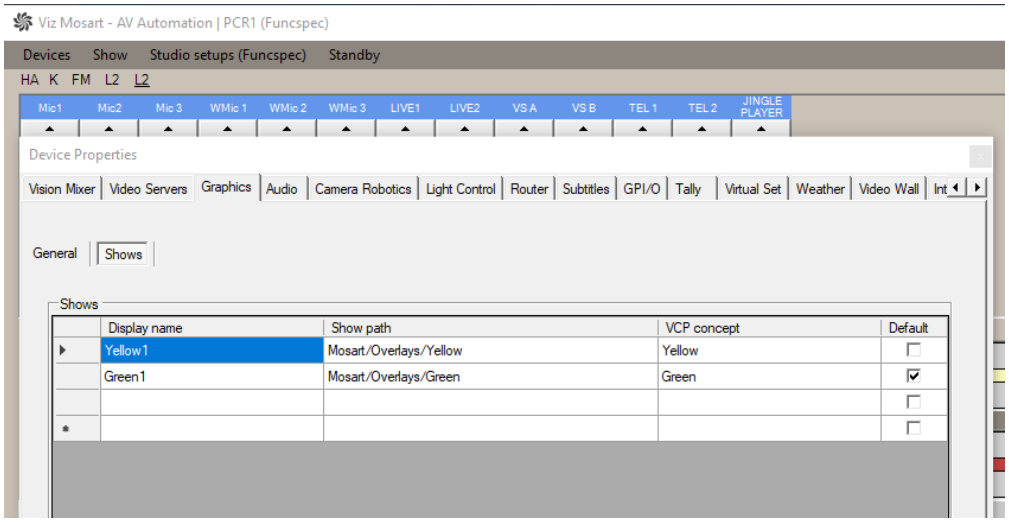
- **Path Externals:** Internal Media Sequencer path to the location for handling database elements. Default: *external/pilotdb/elements*
- **Local Media Sequencer Engine:** Path to the launcher.exe in the Media Sequencer program files folder (only if Media Sequencer is running as a console application on the same machine).
- **Start MSE if not running:** If checked and the Media Sequencer is not running, then AV Automation will try to execute it from the location given below.
- **Use Concept Override:** If the full screen graphics concept should change when a new graphics Profile is selected, this is configured here.
  Select to enable the Vizrt Concept Override if your scenes are compatible with this (requires Viz Pilot 5.2 or later).
- **Use MSE Playable Cache**: When enabled, the *playable_cache* plugin in the Media Sequencer (MSE) will be used. For further details on how the setting should be used, please contact Vizrt Support. This setting requires MSE *5.4 or higher.*
- **Translate Cue for Transition Logic elements:** Only applicable when using Transition Logic full screen graphics as the Media Sequencer ignores a normal cue command for these items.
  - **No translate**: Send cue command to the Media Sequencer (on Media Sequencer versions below 1.20 no cue will be performed).
  - **Take**: Use a take command when cuing.
  - **Take out**: Use a take out command when cuing.
- **Apply Channel Name to Elements:** This will send the Channel name to be displayed with the graphic elements in the GUI.
- **Take out full screen graphics:** Check to take out full screen graphics. By default, Viz Mosart will not do a takeout for full screen graphics.

Advanced settings (not shown on UI)

- **ResetFullscreenGraphicsOnReload:** Enable/disable resetting of graphical resources at rundown load/reload (Default: *true*, which will reset graphics).
  When a rundown is loaded/reloaded, by default, graphics-related resources are *reset* to an initial state. This action is performed on any playlist used for fullscreen graphics, and for the output on connected Viz Engines.
  - This advanced setting is found in AV Automation settings, opened with **Ctrl+Shift+S**.

Shows Tab

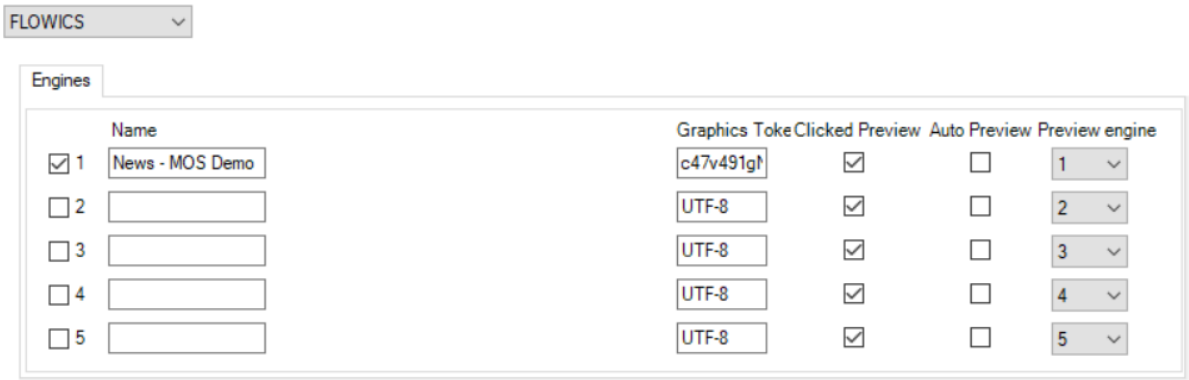
These additional details only appear when VIZRT graphics are selected.



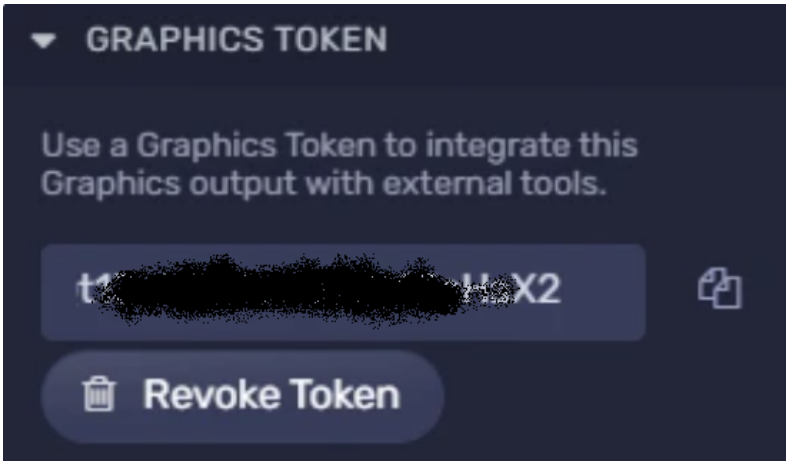
It is mandatory to have an exclusive (non-null) value for **Show path** configured!

Viz Flowics

Viz Flowics provides powerful, customizable, live data-driven HTML graphics.



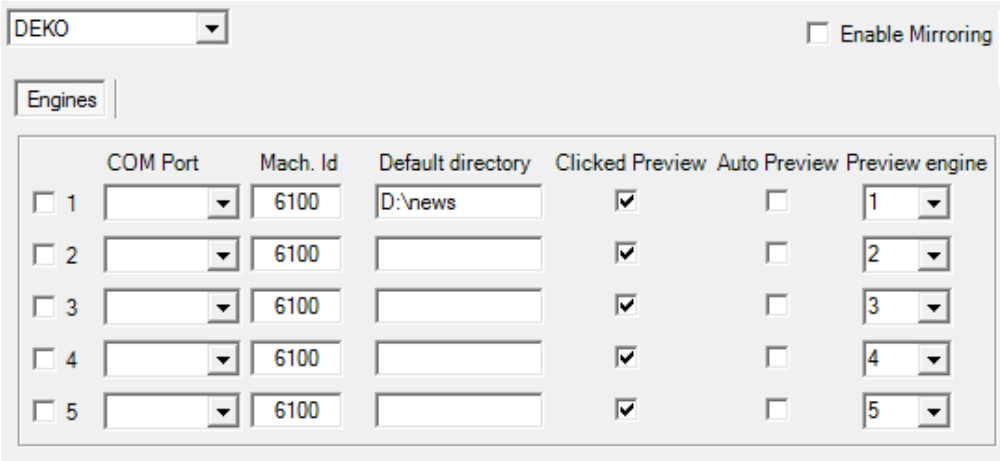
- **Name:** The name of the Graphics package.
- **Graphics Token:** The token generated for the given Graphics package.



You need to install and configure the Viz Flowics MOS Gateway for this integration to operate.

Tip: Follow this Viz Flowics support article for more details on setting up this integration.

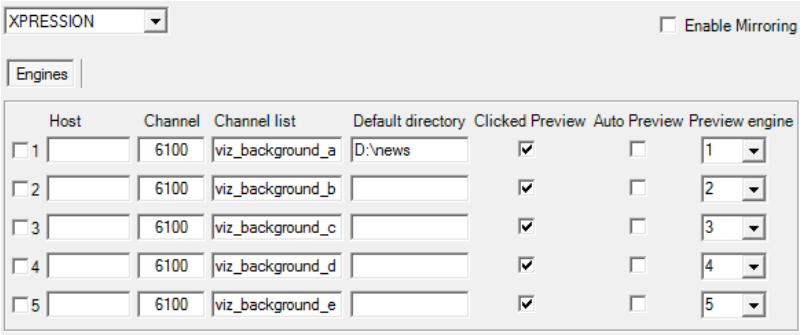
Deko Settings



- **Engines:** Enables the connections to the Deko engines. Viz Mosart supports five graphics engine connections when checked.
- **COM port:** Serial ports connected to the Deko engines.
- **Mach. Id:** Machine IDs of the Deko engines. Valid range is `0` to `9`. Leave blank to send global commands.
- **Default Directory:** The folder on the Deko engine that contains the graphics templates.
- **Clicked Preview:** Click on full-screen graphic element to send to the Preview engine.
- **Auto Preview:** Automatically send full-screen graphic element to the Preview engine.
- **Preview engine:** Defines the Deko engine number.

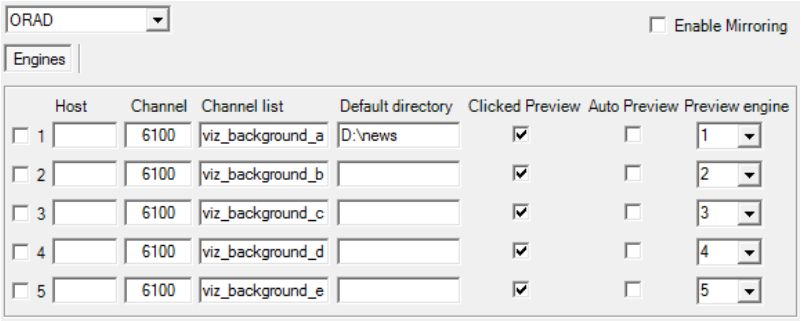
Copyright © 2026 Vizrt

XPression Settings



- **Engines:** Check to enable the connections to the XPression engines. Viz Mosart supports five graphics engine connections.
- **Host:** Hostname or IP address to the XPression engine.
- **Channel:** Output channel from the XPression engine. Use `0` for default channel.
- **Channel list:** Output channel(s) to be cleared when Channel is set to 0.
- **Default directory:** The folder on the XPression engine that contains the graphics templates.
- **Clicked Preview:** Click on full-screen graphic element to send to the Preview engine.
- **Auto Preview:** Automatically send full-screen graphic element to the Preview engine.
- **Preview engine:** Defines the XPression engine number.

Orad Settings

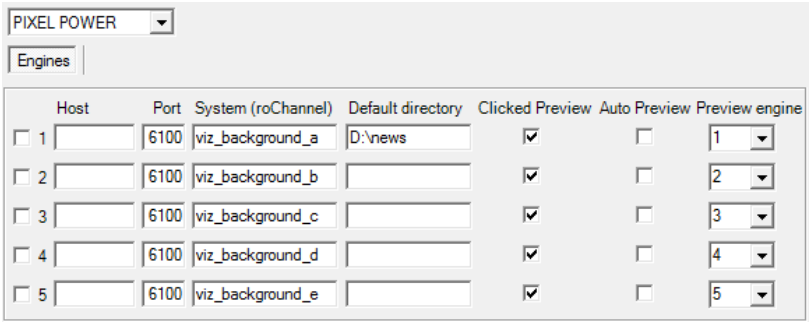


- **Engines:** Check to enable the connections to the Orad engines. Viz Mosart supports five graphics engine connections.
- **Host:** Hostname or IP address to the Orad engine. Include the IP port number, for example 172.20.51.55:10001.
- **Channel:** Output channel from the Orad engine. Use `0` for default channel.
- **Channel list:** Output channel(s) to be cleared when Channel is set to 0.
- **Default directory:** The folder on the Orad engine that contains the graphics templates.
- **Clicked Preview:** Click on full-screen graphic element to send to the Preview engine.

Copyright © 2026 Vizrt

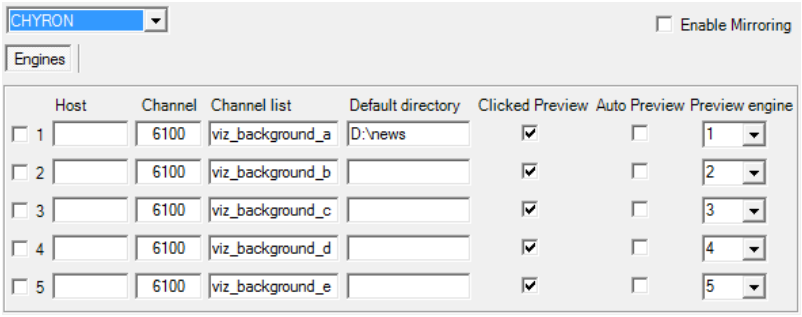
- **Auto Preview:** Automatically send full-screen graphic element to the Preview engine.
- **Preview engine:** Defines the Orad engine number.

Pixel Power Settings



- **Engines:** Check to enable the connections to the Pixel Power engines. Viz Mosart supports five graphics engine connections.
- **Host:** Hostname or IP address to the Pixel Power engine.
- **Port:** Port to communicate with the Pixel Power engine.
- **Default directory:** The folder on the Pixel Power engine that contains the graphics templates.
- **Clicked Preview:** Click on full-screen graphic element to send to the Preview engine.
- **Auto Preview:** Automatically send full-screen graphic element to the Preview engine.
- **Preview engine:** Defines the Pixel Power engine number.

Chyron Settings



- **Engines:** Check to enable the connections to the Chyron engines. Viz Mosart supports five graphics engine connections.
- **Host:** Hostname or IP address to the Chyron engine.
- **Channel:** Output channel from the Chyron engine.
- **Channel list:** Output channel(s) to be cleared when Channel is set to 0.
- **Default directory:** The folder on the Chyron engine that contains the graphics templates.
- **Clicked Preview:** Click on full-screen graphic element to send to the Preview engine.

Copyright © 2026 Vizrt

- **Auto Preview:** Automatically send full-screen graphic element to the Preview engine.
- **Preview engine:** Defines the Chyron engine number.

5.4.4 Vizrt Graphics

For Vizrt graphics, the workflow in Viz Mosart is continually being improved.

- The Viz Mosart integration towards other Vizrt products is now handled by a single driver that communicates with Vizrt's Media Sequencer over a REST API exposed by Media Sequencer.
- In future versions, a expanding range of both existing and new Viz playout content will be supported.

To configure and use the new Vizrt Graphics driver

- Please refer to section *Vizrt Graphics settings*.

Copyright © 2026 Vizrt

5.4.5 Audio

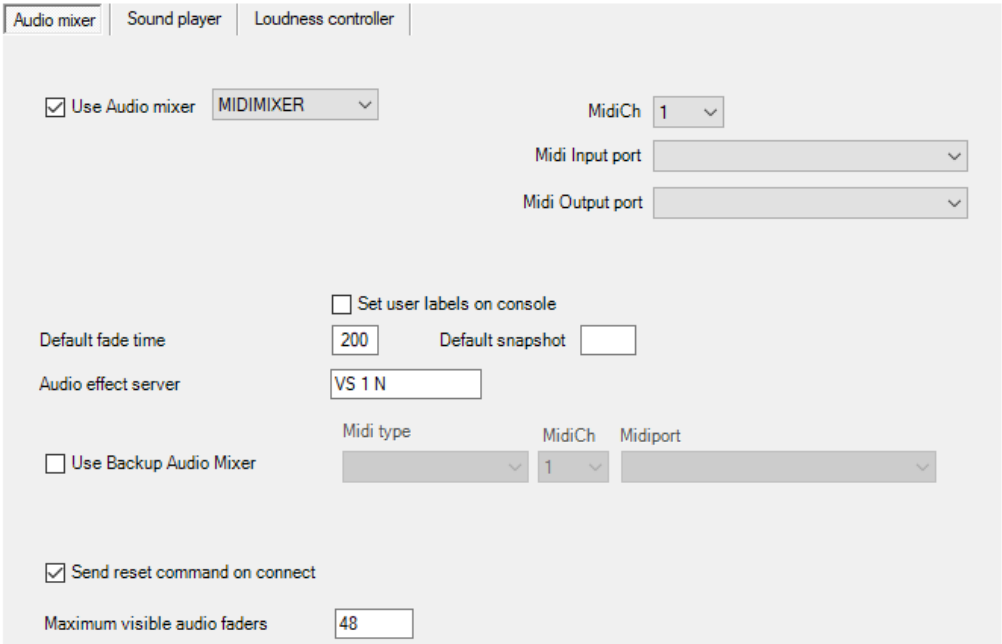
Viz Mosart connects to a variety of audio devices, from various manufacturers.

Select the required audio device from **AV Automation > Devices** > **Preferences** > **Audio**.

- Audio Mixer
  - Audio Mixer Types
  - AV Automation Audio Mixer Availability Table
- Audio Players
- Loudness Control

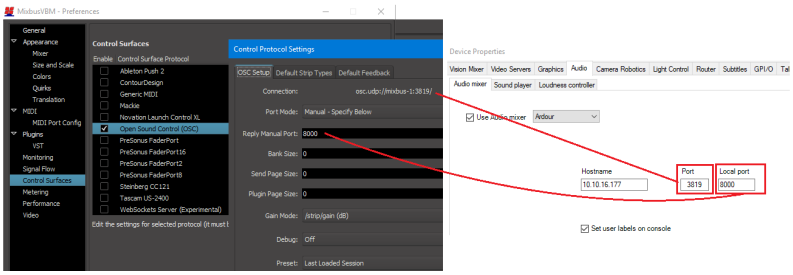
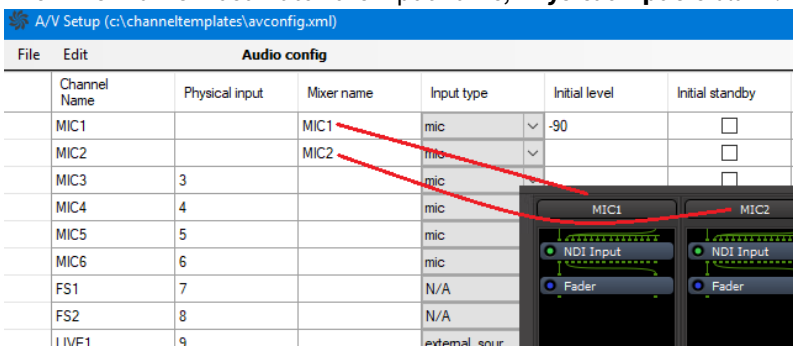
Audio Mixer

This tab enables setting-up of the main mixer and optionally, a backup.



- **Use Audio mixer:** Enables the audio mixer functionality.

Audio Mixer Types

Audio mixer	Protocol	Supported hardware / Comments	Interface
Ardour / Harrison Mixbus VBM	OSC	Harrison Mixbus VBM • Ensure *OSC* is enabled in the **Control Surface Protocol** selector and that in the **OSC Setup** tab, Viz Mosart Audio mixer properties match:  In the example above, where the port number (here *3819*) match correctly. • The input field **Port Mode** <u>must</u> be: *Manual - Specify Below*. • The input field **Reply Manual Port** can be any value, including empty. • The **Mixer name** must match the input name, **Physical Input** is blank:  **Info**: Harrison MixBus 32C is *not* supported.	UDP
CALREC	RAP		RS-422, TCP/IP (Artemis)
DHD	DHD RM4200-D v2.02		TCP/IP

Audio mixer	Protocol	Supported hardware / Comments	Interface
EUPHONIX	RAP		COM
LAWO EMBER+	Ember+	Lawo audio mixers with A__UHD Core. This requires that the Lawo audio mixer has correct firmware and settings to allow Ember+ control (contact Lawo for details about which specific audio mixer models support this functionality).	TCP/IP
LAWO RMNOPL	Remote MNOPL	Lawo mc²	TCP/IP
LAWO ZIRKON	RAS, Lawo Zirkon subset		RS-422 or TCP/IP
MIDIMIXER	Sanford. Multimedia		MIDI
SSL	Light Broadcast Automation protocol, SSL Broadcast Automation Control Specification – v1.0		RS-422 (for SSL over TCP/IP, select CALREC Audio Mixer)
STAGETEC	Stagetec Diamond RAS protocol, Monitora RAS		COM

Audio mixer	Protocol	Supported hardware / Comments	Interface
STAGETEC EMBER+	Ember+	Unless advised otherwise, try 9000 for Port.	TCP/IP
STUDER_3000	Monitora + Ember, Ember 1.0	Studer OnAir 3000	RS-232 or TCP/IP
STUDER_VISTA	Ember	Studer Vista	TCP/IP
TASCAM SONICVIEW 16 EMBER+	Ember+	Unless advised otherwise, try *9000* for Port.	
TRICASTER (previously vizrt-ips)	NewTek NDI	NDI/NewTek IPseries mixers. TriCaster, TriCaster Vectar, VMC. See also: TriCaster and Viz Mosart.	TCP/IP
VIZENGINE	n/a	Viz Opus only	TCP/IP
WHEATSTONE	Wheatstone Mixer Automation Protocol v.1.4		TCP/IP
YAMAHA_O2	MIDI	Yamaha 01V96, DM1000, DM2000, DME64N, O2V96, LS9, CL5	MIDI

- **Serial (Calrec, Lawo, Studer, etc.):** Serial port connected to the audio mixer, when **Serial** radio button is selected.

- **TCP/IP:** Hostname or IP address and port to communicate with the audio mixer, when **TCP/IP** radio button is selected. Note that for *Calrec* it is possible to configure a main and backup audio mixer by specifying two comma separated hostname values and port values.
- **Hostname:** Hostname or IP address of the audio mixer.
- **Port:** Port on the audio mixer.
- **Local port:** (Ardour only) The mixer will send messages to this port on the Mosart server. Use 8000 unless there is a good reason for choosing another port.
- **Backup hostname and port (LAWO RMNOPL):** Hostname (or IP address) and port to the backup audio mixer console.
- **Set user labels on console:** When selected, user labels on the (non-EMBER+) Lawo or Vista console are overwritten.
- **MidiCh (MIDIMIXER and YAMAHA_O2):** MIDI channel assigned to the audio mixer. Depending on mixer model, this setting may have to correspond to settings on the mixer.
- **Midi Input port (MIDIMIXER and YAMAHA_O2):** System MIDI input port to communicate with the audio mixer.
- **Midi Output port (MIDIMIXER and YAMAHA_O2):** System MIDI output port to communicate with the audio mixer.
- **Default fade time:** Time in frames for fading open faders with the manual fade function (**CTRL+F**).
- **Default snapshot:** Snapshot to recall when starting the automation (MIDI parameter change).
- **Audio effect server:** Name for the Audio effect fader. This fader will open if effect transitions are used.
- **Send reset command on connect:** Used for STUDER3000 SW 2.2, or lower. Recalls a default setup for the Mixer panel.
- **Visible audio faders:** Maximum visible faders in AV Automation.

AV Automation Audio Mixer Availability Table

	Serial	TCP/IP	MidiCh	Midi port	Hostname	Port	Backup host name	Backup Port	Fader input offset	Usee mber protocol	Comport audio mixer
ARDOUR					X	X					
CALREC	x	x			x	x					x
DHD		x			x	x					
EUPHONIX	x										x
LAWO RMNOPL		x			x	x	x	x			

	Serial	TCP/IP	MidiCh	Midiport	Hostname	Port	Backup host name	Backup Port	Fader input offset	Useember protocol	Comport audio mixer
LAWO ZIRKON	x	x			x	x					x
LAWO EMBER+					x	x					
MIDI MIXER			x	x							
SSL	x								x		x
STAGE TEC	x	x			x	x					x
STAGE TEC EMBER+					x	x					
STUDER 3000	x	x			x	x				x	x
STUDER VISTA		x			x	x					
TASCAM SONIC VIEW 16 EMBER+					x	x					
VIZ ENGINE		x			x	x					

	Serial	TCP/IP	MidiCh	Midi port	Hostname	Port	Backup host name	Backup Port	Fader input offset	Useember protocol	Comport audio mixer
WHEATSTONE		x			x	x					
YAMAHA O2			x	x							
TRICASTER (previously vizrt-ips)		x			x	x					

Audio Players

- Viz Mosart Audio Player
- TriCaster Audio Player
- Third Party Audio Players

You select from available audio players at **AV Automation > Devices > Properties > Audio** tab **> Audio Player** sub-tab.

Viz Mosart Audio Player

Requirements

Refer to general section Prerequisites.

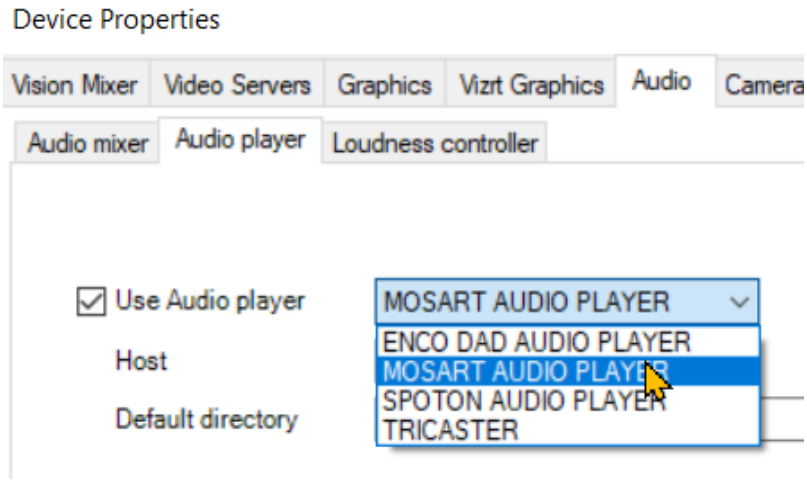
Supported File Formats

The Viz Mosart Audio Player supports formats: mp3, wav, wma, aac, m4a, mp4, aiff, avi.

To set up the Mosart audio player

1. On the machine that will be used for the audio player, open the Viz Mosart installation folder and locate the Audio Player installer file, named something like VizMosartAudioPlayer-5.13.2.32554.msi
2. Install the audio player, ensuring to add a Desktop shortcut, for user convenience.
3. Start the program from the Desktop.
4. On the Viz Mosart server, open AV Automation.
5. Go to **Devices** > **Preferences** > **Audio** > **Audio player.**

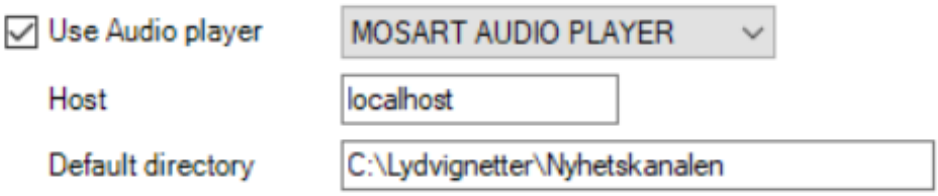
6. Check **Use Audio player** and choose from the players in the drop-down list.



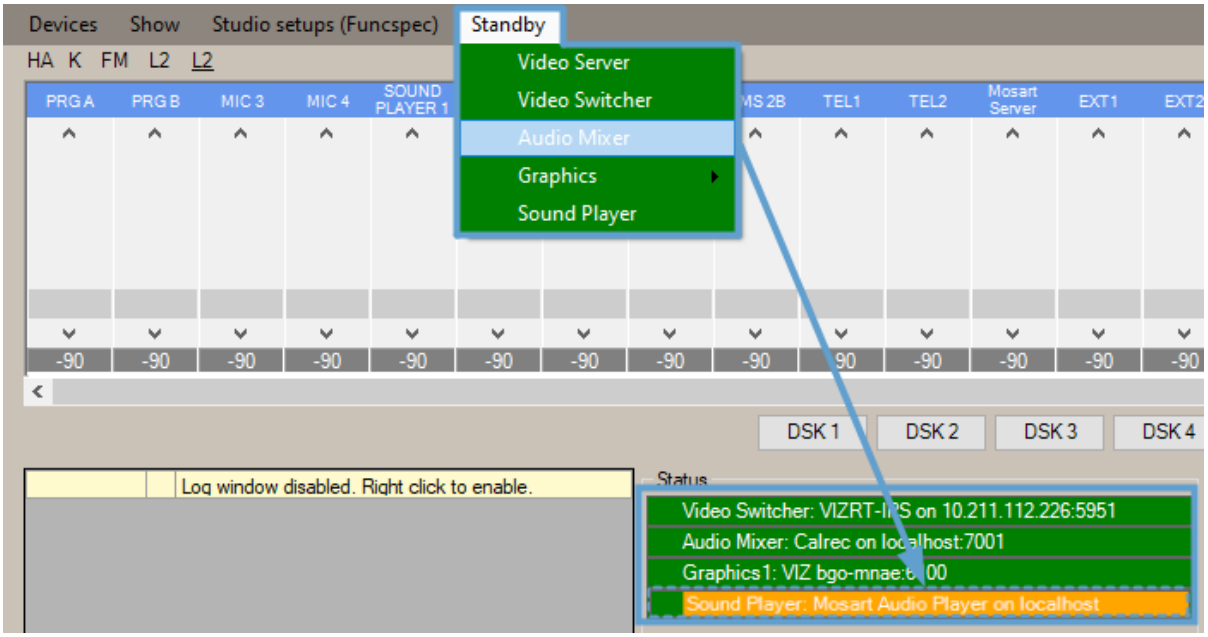
7. Under **Host** write the hostname or IP address of the computer where the audio player is running (the machine selected in step **1**. above).

Info: To select alternative port usage, see Audio Player Port Numbering below.

7. Under **Default directory**, write the folder path where the audio files are stored. This is specified by the setting **Audio Directory** in the Viz Mosart Audio Player.



8. Restart AV Automation and check that the connection is green.



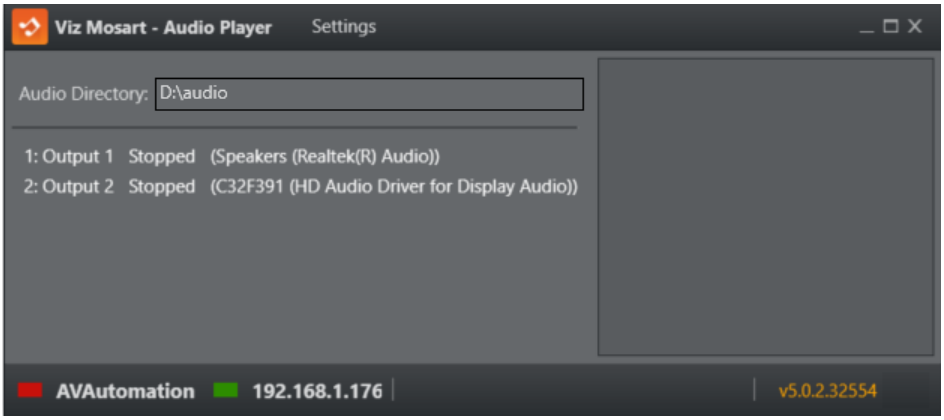
Working with the Viz Mosart Audio Player

When the Viz Mosart Audio Player is configured, the user opens the UI from a desktop shortcut

or file location “%programfiles(x86)%\Mosart Medialab\Mosart Audio Player\Mosart Audio Player.exe”.

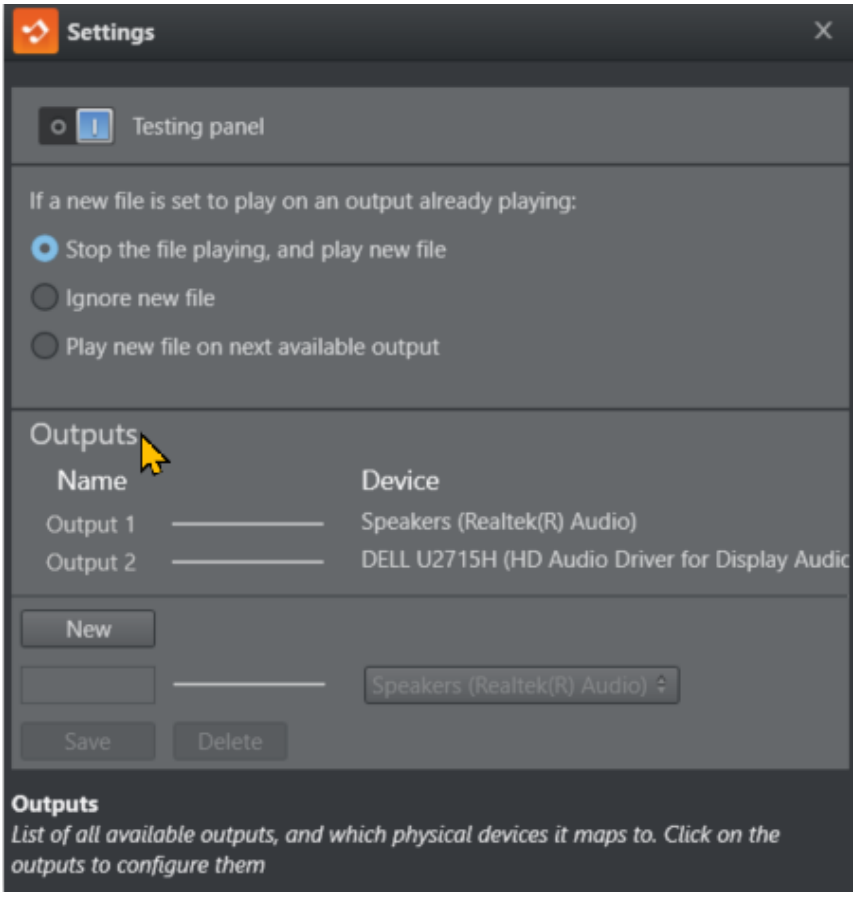
Info: User actions are described in section *Playing Audio* of the *Viz Mosart User Guide*.

Viz Mosart Audio Player UI



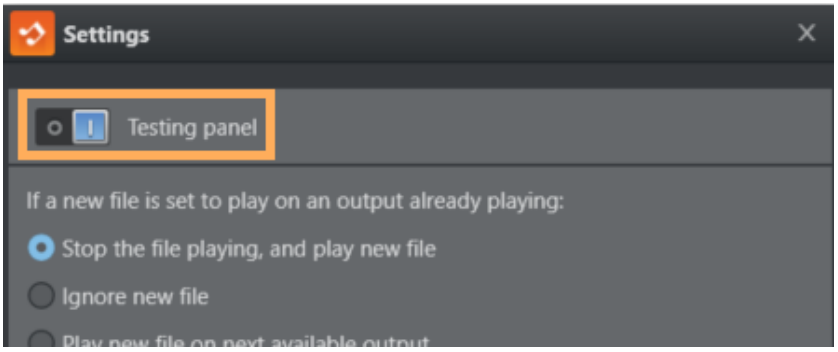
UI Element	Description
Settings	Open the Settings menu for configuring the Audio Player.
Audio Directory	Folder path to available audio files. All files in this folder are ready to be played This location must be the same as is set as **Default directory** in AV Automation **Devices > Properties > Audio > Audio Player**. Only files located in this directory are available to the Viz Mosart audio player.
Numbered Output list	A list of all available devices. The numbered entity under **Name** is the number you use in AV Automation or the NRCS to specify the audio file playout (for example, in the screenshot above, *Output 1*).
Log panel (right hand panel)	All log messages that the audio player creates.
AVAutomation (status)	**Green**: AV Automation is connected **Red**: No AV Automation connection.
IP Address	IP address of the computer that the Audio Player is running on.
Version	Version number of the Audio Player.

The Settings menu



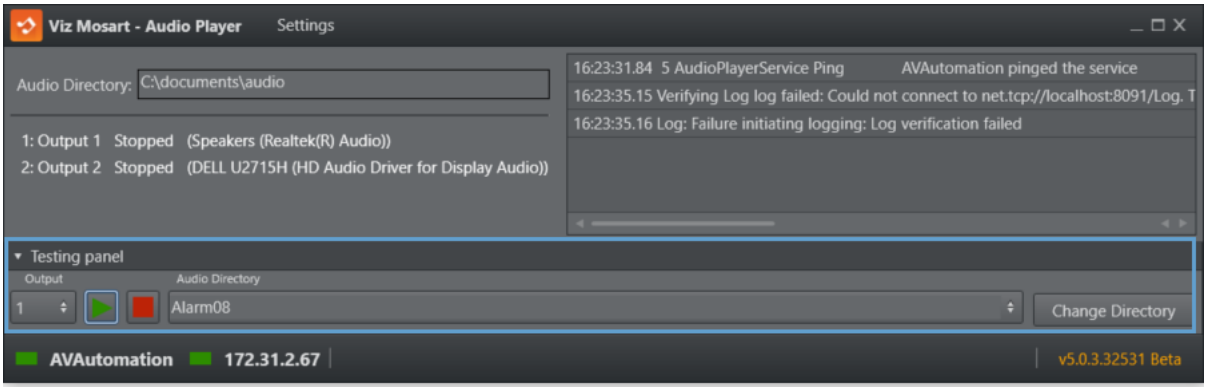
Testing panel

- Select this option to display a player at the foot of the Audio Player panel.



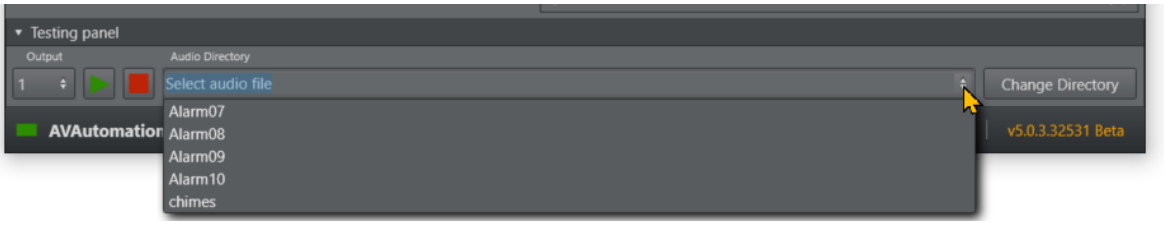
To select the directory containing your audio files

- Click **Change Directory**.

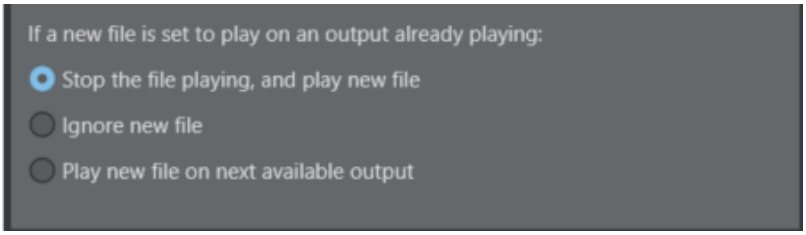


To select a file for playing

- Click the double-arrow to list and select a file.



Behavior when file is already playing



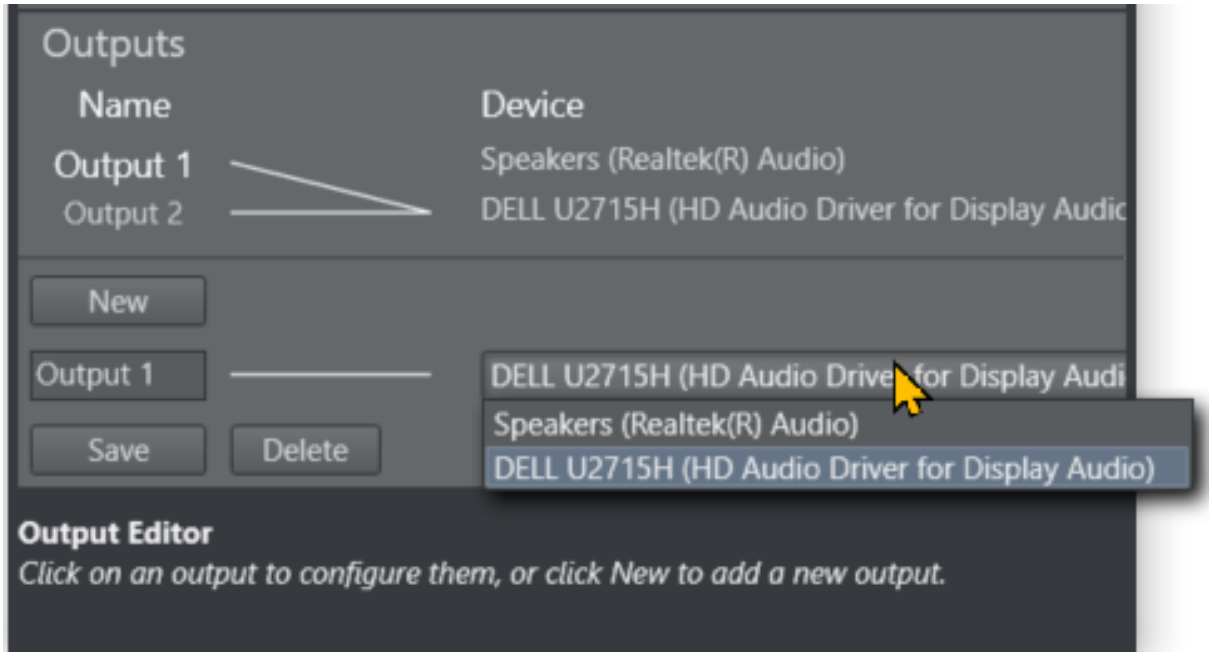
- In this panel, you can determine what action the Audio Player will take when a new file is played and a file is already playing.

Outputs

The **Name** column shows all outputs. The **Device** column shows all available playback devices on the computer. The line between them shows which playback device the output uses to play the audio file.

To select audio device for an output

1. Click an Output. (In this case, Output 1 has been clicked.)
   The panel below displays a drop-down menu of available devices.

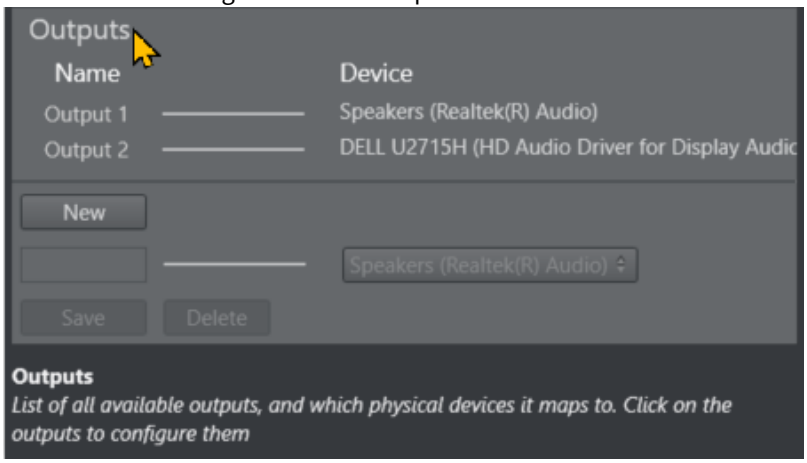


2. Choose from the displayed devices to associate one of your computer's playback devices with the selected output.
3. (Other operations) Here you can also
   a. **Rename**
      Overwrite the content of a name under **Outputs** (for example *Output 1* above) by first clicking on it, typing a new name, then clicking **Save**.
   b. **Delete**
   c. Create a **New**
      output.

Help

When working with settings, a tooltip box at the bottom of the panel displays additional information about the current operation.

- Mouse-over a setting to see the tooltip.



Audio Player Port Numbering

The port number used by the Audio Player is by default *8084*.

To change Audio Player port number

If the Player port number conflicts with some other service, it can be changed.

1. Editing the Audio Player Configuration file *Mosart Audio Player.exe.config* located at `%ProgramFiles(x86)%\Mosart Medialab\Mosart Audio Player\`.
2. Change the *baseAddress* attribute to use another port, for example 8087.
   `<add baseAddress="http://localhost:8087/LiveSoundService.svc/"/>`
3. If the port number is changed in this way, the AV Automation settings for the Audio Player must also be modified to use this port number.
   Specify the port number in the **Host** field in AV Automation's Audio Player properties to this format: *Server=localhost;Port=8087;*
4. To verify that the Audio Player actually uses the new port, you can test if the URL (for example, *http://localhost:8087/LiveSoundService.svc/*) is responding in a web browser.

Operating the Audio Player

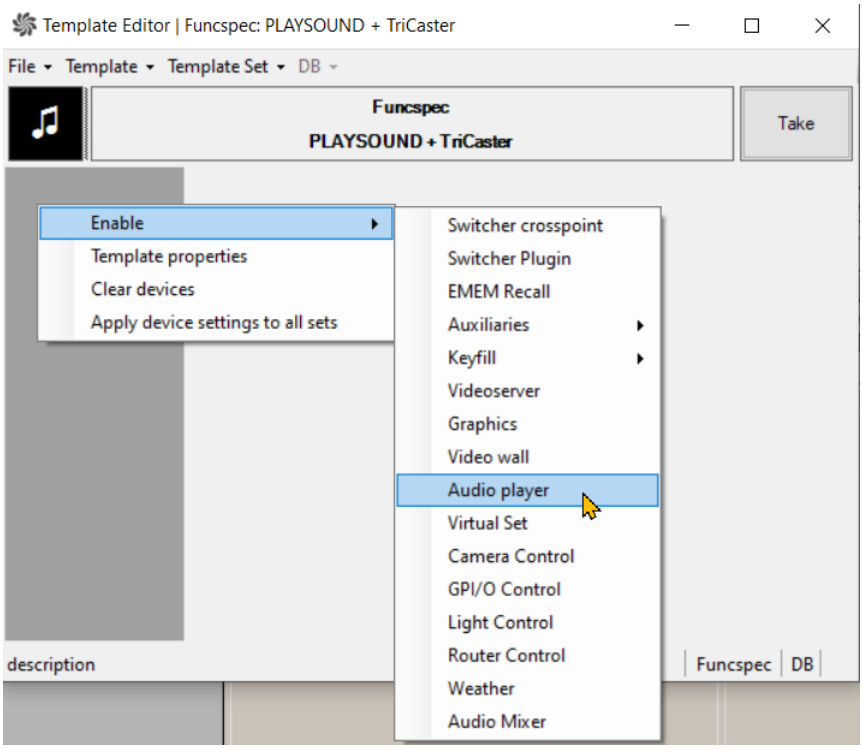
Refer to section *Playing Audio* in the *Viz Mosart User Guide*.

TriCaster Audio Player

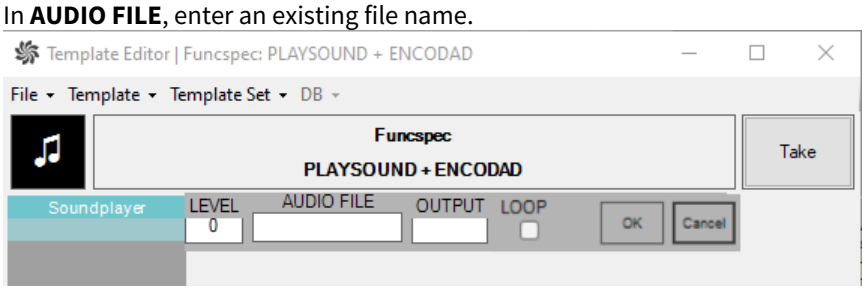
By selecting *TriCaster* from the list of available audio player devices, you can control (access and play audio clips) the TriCaster onboard audio player from a Viz Mosart template.

To set up a TriCaster audio player

1. Start the TriCaster on a host machine available to the Viz Mosart server.
2. In AV Automation, navigate to **Devices > Properties > Audio** (tab) **> Audio player** (tab).
   a. From the drop-down menu to the right of **Use Audio player**, select *TriCaster* (previously called *vizrt-ips*).
   b. In the **Host** field, enter the hostname or IP address of the TriCaster host.
   c. Choose SpotOn Audio Player In **Default directory**, enter the audio clips directory on the TriCaster machine (for example, C:\ProgramData\vizrt\Media\Sound and Music\Sample\www.SoundTraxxMusic.com).
   d. Click **Save & Close**.
3. Restart AV Automation and make sure you have a *green connection* (as in the SpotOn audio player example below).
4. In AV Automation, navigate to **Template Editor > Template > Select** or **Template Editor > Template > New**.
5. Choose or create an appropriate template, right-click in the dark grey area to the left and select **Enable > Audio player**.

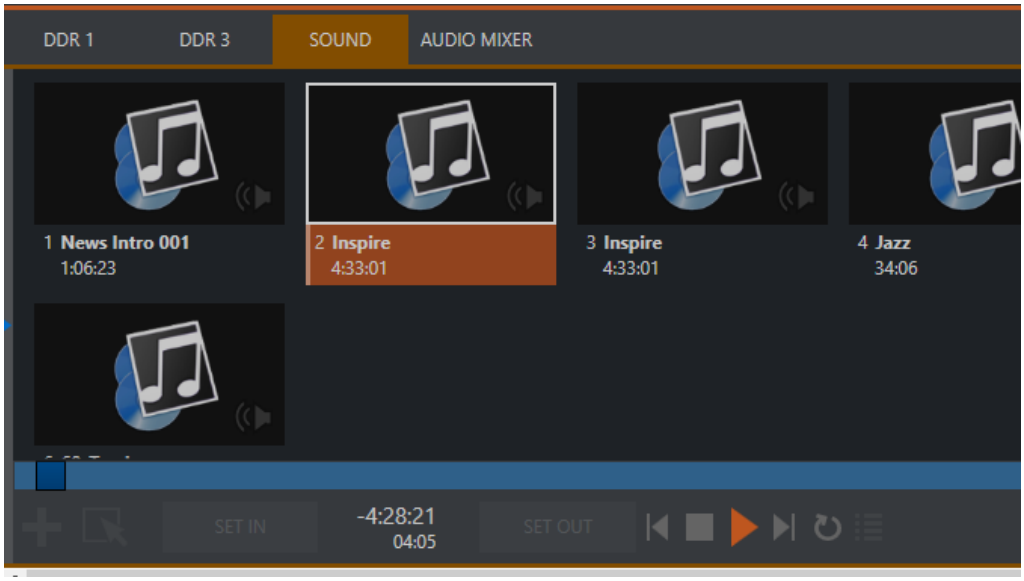


6. In **AUDIO FILE**, enter an existing file name.



> **Info**: Looping (selecting **LOOP**) is not currently available for TriCaster

These can be browsed in the **SOUND** tab on the TriCaster (in this example, *Inspire.mp3*).



7. On the Soundplayer device, click **OK**.

See also: TriCaster and Viz Mosart.

Third Party Audio Players

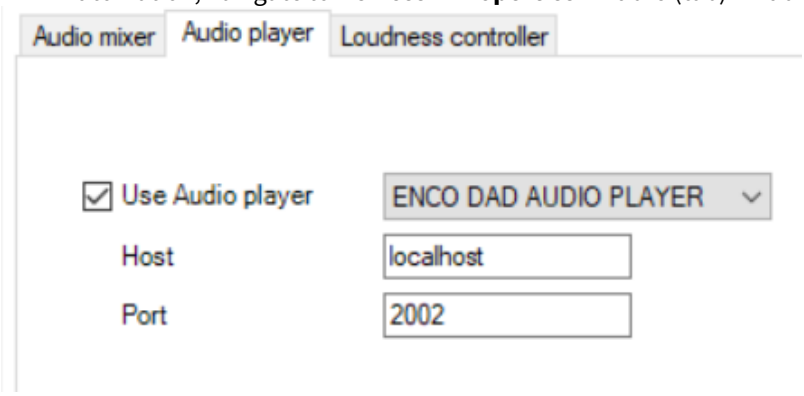
- ENCO DAD Audio Player
- SpotOn Audio Player

ENCO DAD Audio Player

By selecting *ENCO DAD Audio Player* from the list of available devices, you can control (access and play audio clips) the ENCO DAD audio player from a Viz Mosart template.

To set up an ENCO DAD audio player

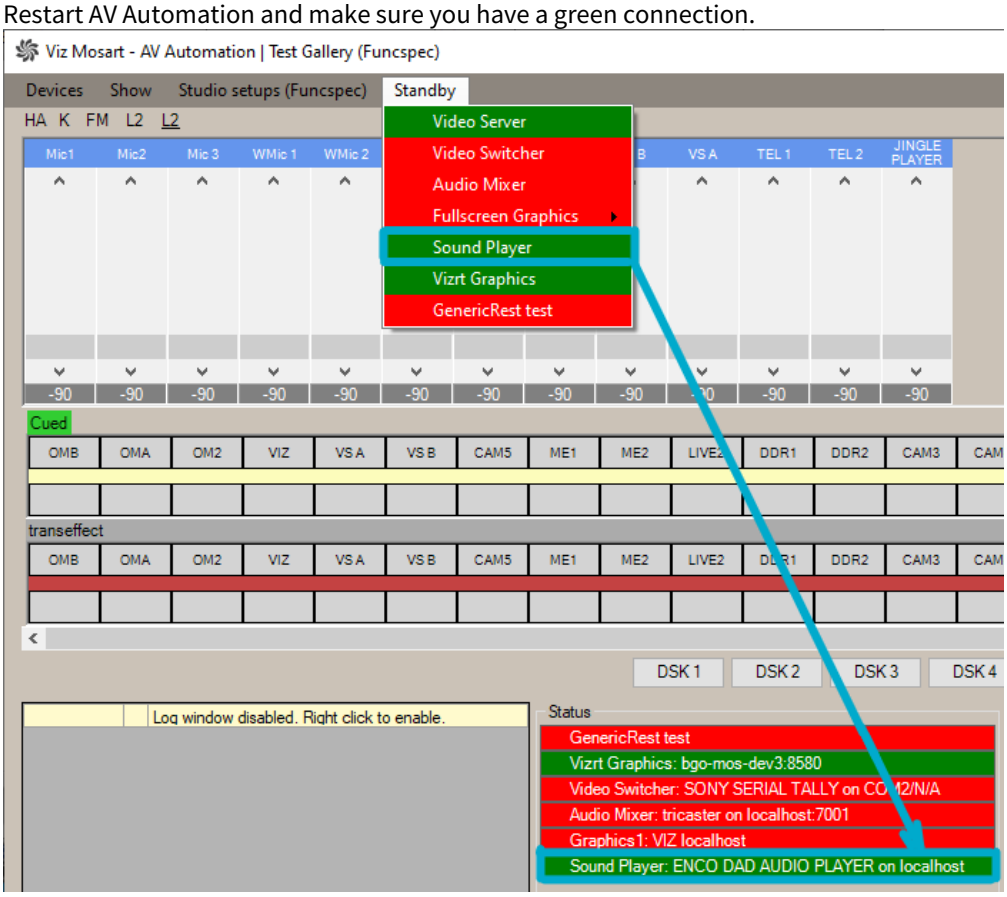
1. Start the ENCO DAD on a host machine available to the Viz Mosart server.
2. In AV Automation, navigate to **Devices > Properties > Audio** (tab) **> Audio player** (tab).



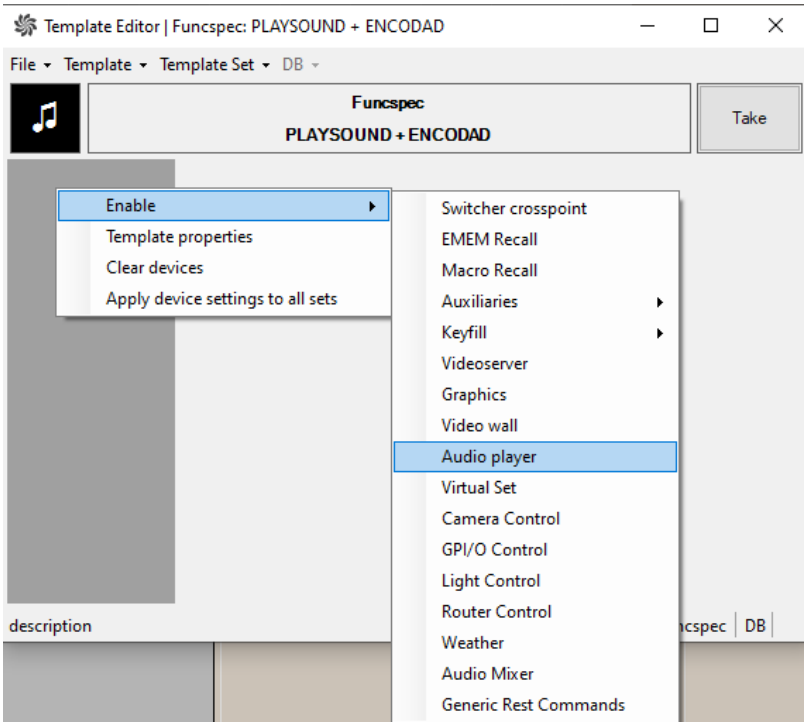
a. Check **Use Audio player**.
b. From the drop-down menu to the right of **Use Audio player**, select *ENCO DAD AUDIO PLAYER*.
c. In the **Host** field, enter the hostname or IP address of the ENCO DAD host.

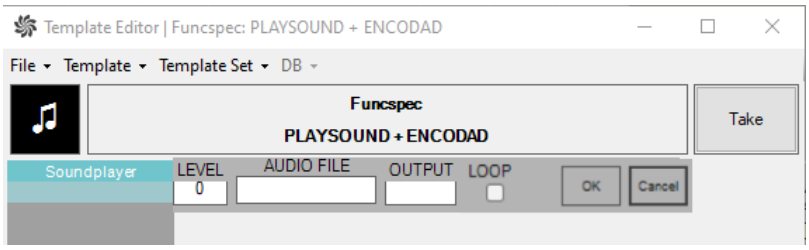
d. In the **Port** field, enter the ENCO DAD monitors port. At clean installation, this value is initially preset as *2002*, but if left empty, clears to *0*.
e. Click **Save & Close**.

3. Restart AV Automation and make sure you have a green connection.



4. In AV Automation, navigate to **Template Editor > Template**.
5. Choose an appropriate template, right-click in the dark grey area to the left and select **Enable > Audio player**.





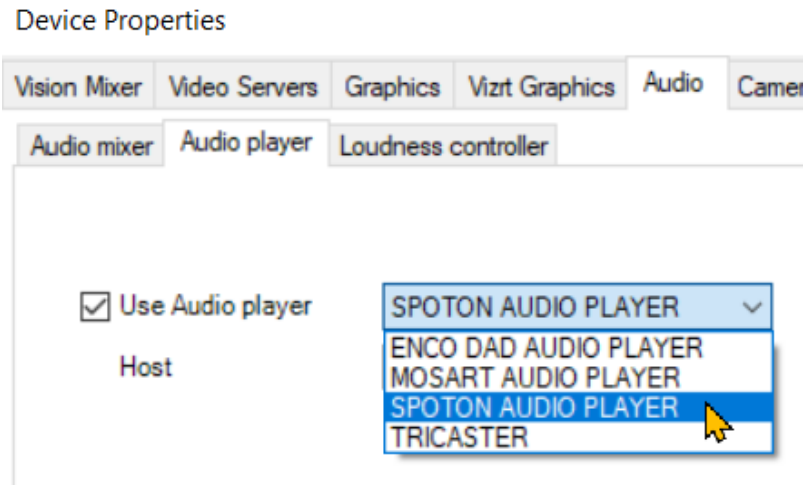
Info: Looping (selecting **LOOP**) is not currently available for Third Party Audio Players.

6. In **AUDIO FILE**, enter the name of a file in the ENCO DAD library.
7. In **OUTPUT**, enter a valid machine number (in ENCO DAD).
8. For **Soundplayer** device, click **OK**.

SpotOn Audio Player

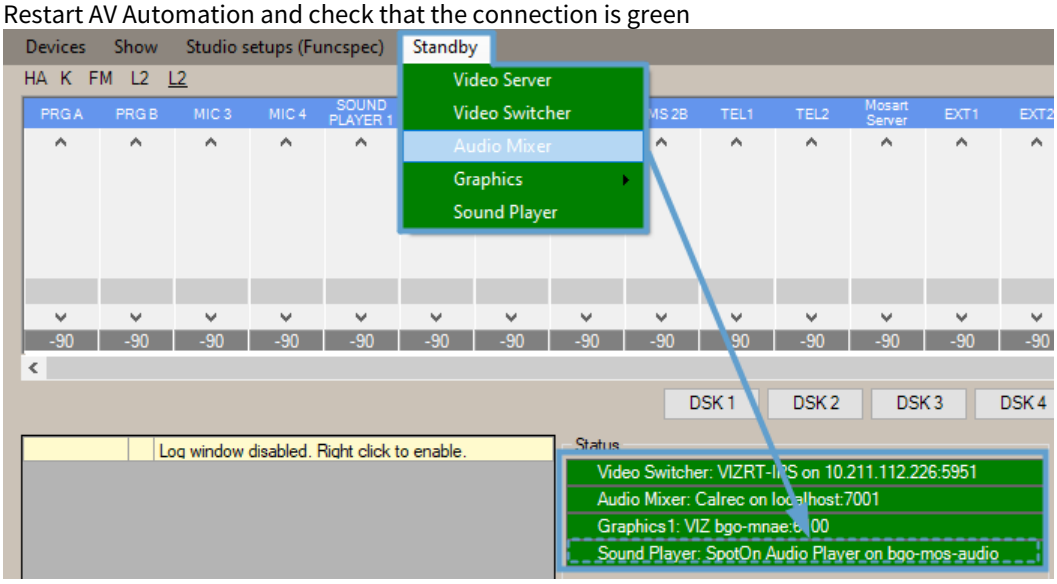
To set up a SpotOn audio player

1. Download, install and start the device software to a host machine available to the Viz Mosart server.
2. In AV Automation, go to **Devices** > **Properties**> **Audio** > **Audio player.**
   a. Check **Use Audio player**.
   b. Choose *SpotOn Audio Player* from the players in the drop-down list.



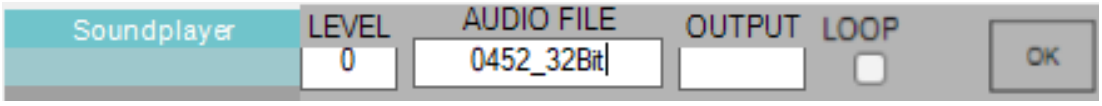
c. Under **Host** write the IP address to the machine where the SpotOn player is running.
d. Click **Save & Close**.

3. Restart AV Automation and check that the connection is green



Working with the SpotOn Audio Player

Addressing an audio file that is stored in a SpotOn client is similar to the default player method above, but the file identifier is different.



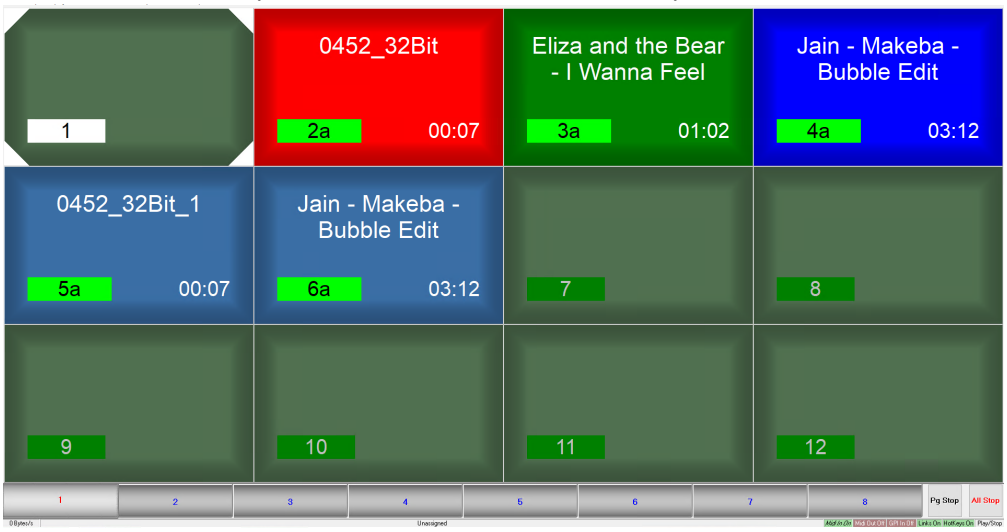
> **Info**: Looping (selecting **LOOP**) is not currently available for the SpotOn player.

- **LEVEL:** This value *does not* affect the audio fader that is taken to Program. Instead a special command option is offered. Set LEVEL to *-90* to send a **STOP** (playing audio) command. All other values are ignored.

- **OUTPUT:** Enter the panel *button number* to identify the associated audio file (in this sample screenshot below, *2a*).

Tip: Use the *button number* (mapped to OUTPUT) as reference (instead of the audio file name). You can then replace the audio file assigned to a button (for example, when a show is rebranded and new music selected) and still keep the same template information.
This avoids having to change the file name.

- **AUDIO FILE:** Enter the panel button name here (in this example *0452_32Bit*).



Info:

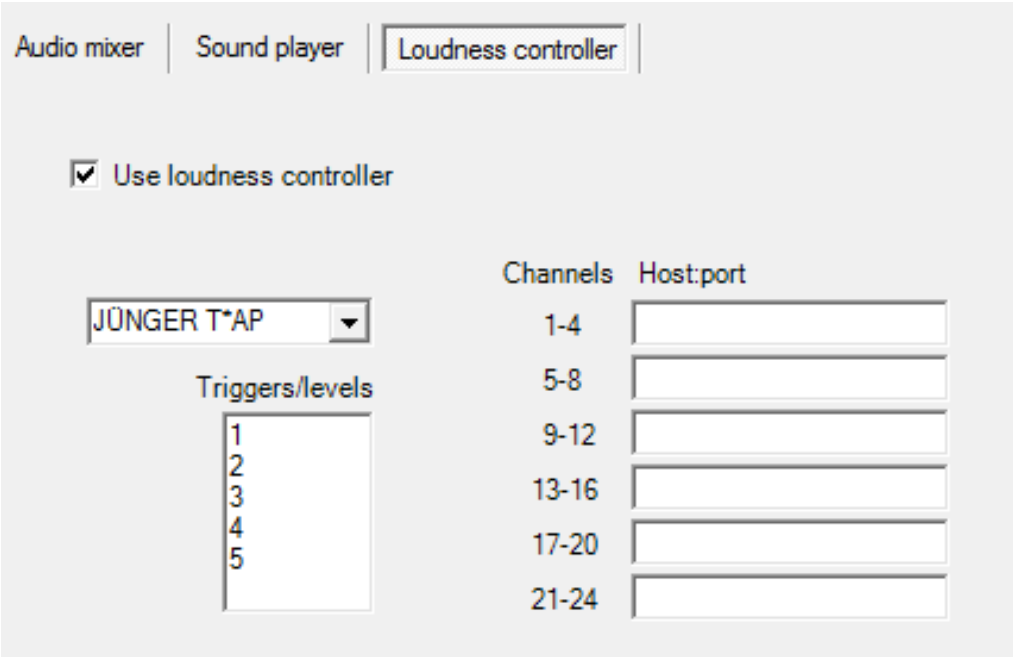
- Newsroom tag available. If a newsroom tag is added, and a value is provided by the NRCS, that value for the file name (with either AUDIO FIE or OUTPUT) is used.
- When both panel button *number* and panel button *name* are used as file identifier, the value of panel button *number* takes precedence.

Loudness Control

Viz Mosart can interact with a loudness controller from Junger.

To configure loudness control

1. Navigate to **AV Automation > Devices** > **Properties**> **Audio** > **Loudness controller**



2. Follow these setup guidelines:
   a. **Use loudness controller:** Check the box to enable loudness control.
   b. **Loudness controller type:** Dropdown menu selector. Choose *JUNGER T*AP*
   c. **Triggers/levels:** Name of triggers or loudness levels, one for each line. These appear in a tooltip menu of Template Editor, and must correspond to the configuration in the loudness controller(s)
   d. **Channels:** Fixed values. For Junger T*AP there are four stereo channels per host.
   Connections to the audio faders are configured in **AV Automation > Devices > A/V Setup > Edit > Audio config >** (**Loudness channel**)
   e. **Host:port:** Hostname or IP address and optional port number for each controller.
3. **Save & Close**.

> **Info:** If the root node name is different from *T*AP* the name must be set in the AV Automation configuration property *JungerRoot* .

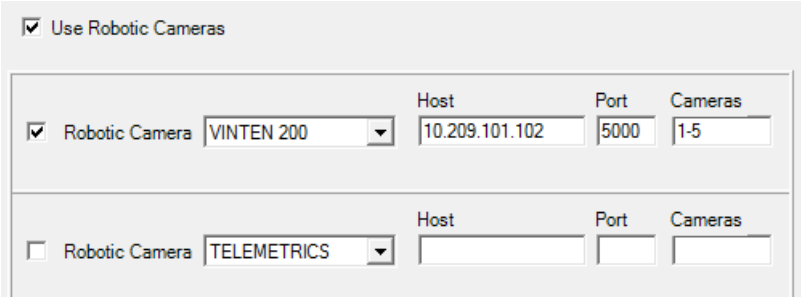
See Also

- Audio Player
- Stagetec Driver Configuration (RAS protocol)
- Audio Mixer Configuration Files

5.4.6 Camera Robotics

Configuring a Robotic Camera

From **AV Automation > Devices > Properties > Camera Robotics tab**, you can configure up to 4 robotic camera controllers.



- **Use Robotic Cameras:** Check to enable the robotic camera control functionality.
- **Robotic Camera:** CAMBOTICS, CAMEROBOT, CINNEO, ELECTRIC FRIENDS, FX-MOTION, NDI PTZ, PANASONIC, PANASONICNEW, PANASONIC TCP, SHOTOKU TR-T, SHOTOKU TR-T IP, TECHNODOLLY, TELEMETRICS, VINTEN 200, .
- **Host:** Hostname or IP address of the *RCC* (Robotic Camera Controller) server.
- **Hosts (FX-Motion, Technodolly and NDI PTZ):** A comma separated list of names or IP addresses for the camera robots starting from Camera 1. Missing robots are indicated by extra commas.
  For NDI PTZ, use NDI Studio Monitor or Viz Mosart UI (see the Viz Mosart User Guide, section *Viz Mosart User Interface > Program Window*) for discovering NDI PTZ cameras in the network.
  The source names displayed there shall be be set in this **Host** field. Use the **External Sources** tab in the NDI Access Tool to reach the cameras on subnets that are not on your current network.
- **Port:** Port to communicate with the RCC server.

Note: When using CAMEROBOT, enter only the hostname or IP address in the **Host** field. Any additional connection-string parameters typed into this field (for example `SendDuration=true`) are discarded before the driver is created, and have no effect. Camerobot-specific settings are made in `RoboticCameraConfiguration.xml`, see Robotic Camera Configuration Files.

Note: When using PANASONIC TCP, you may or may not specify a port here. If no port is specified, the default port 80 is used.

- **Com port:** For serial communication with the RCC server.

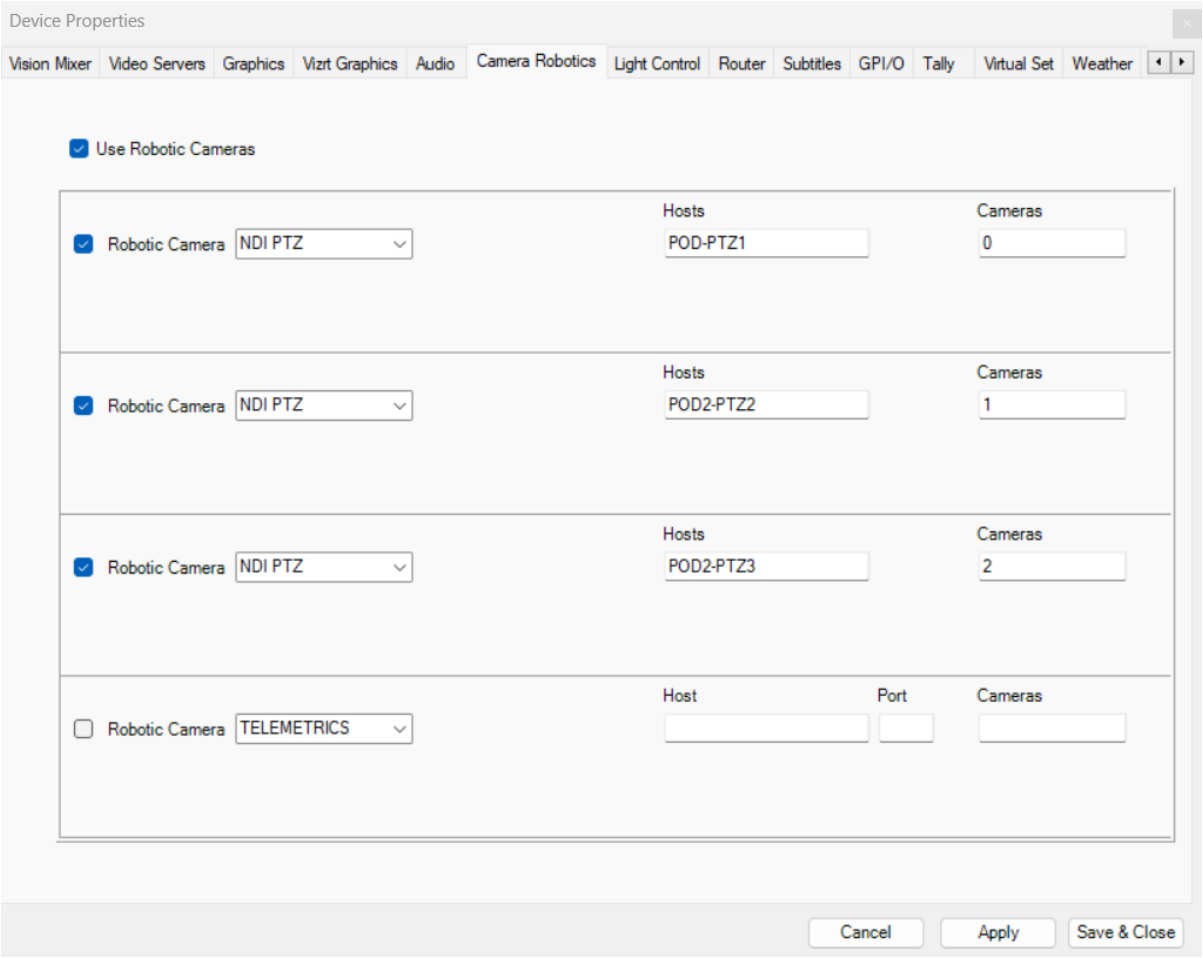
Note: When using PANASONIC or PANASONICNEW you *must* select the Com port to use. Only one controller can be connected to each Com port.

- **Cameras:** List of robotic cameras for individual stand-by control and for matching against the Camera Number in Robotic Camera Control in Template Device Functions. List members are separated by commas, or by hyphens indicating ranges.

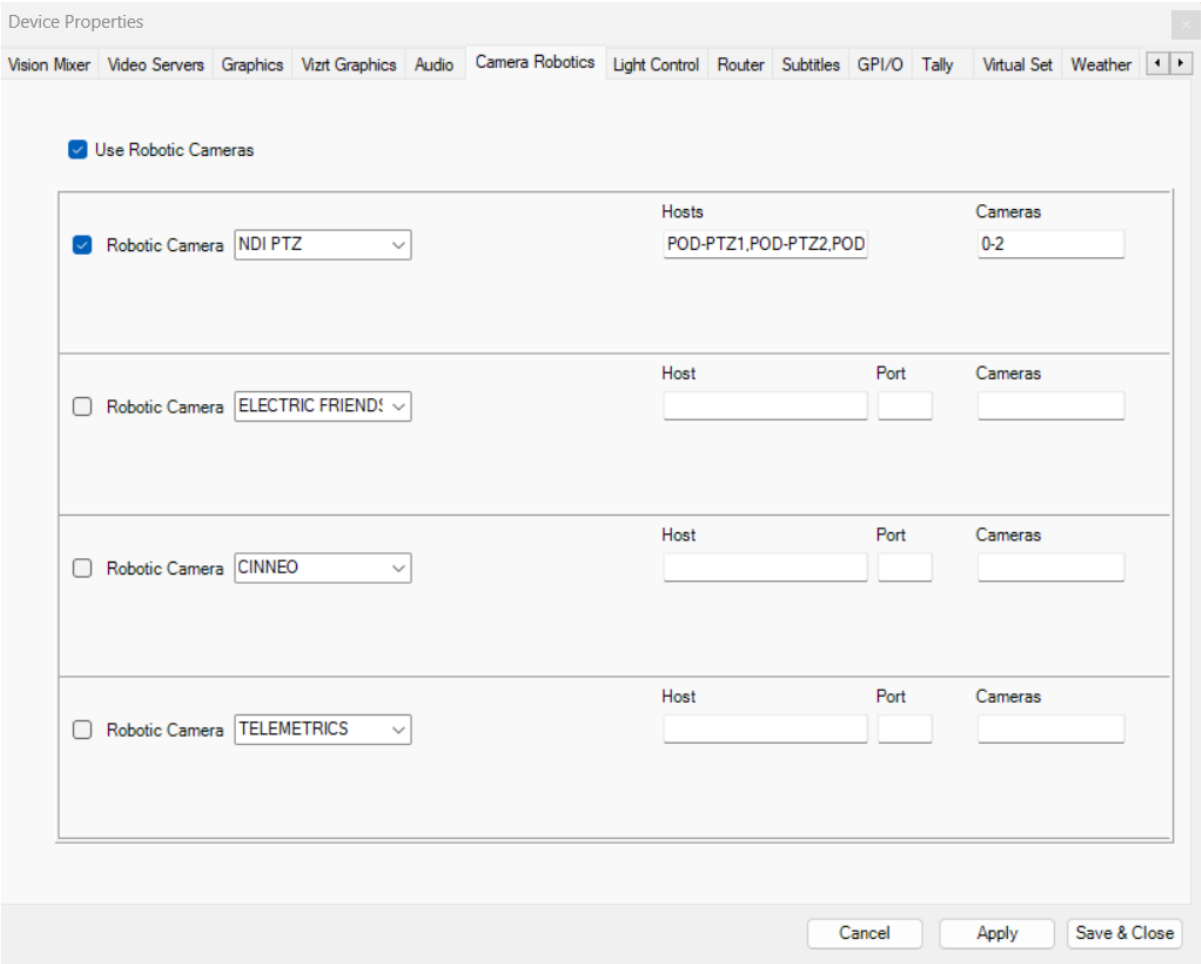
> **Note:** When using PANASONIC, PANASONICNEW, or PANASONIC TCP, the connection goes through a controller. The *Cameras* value is used to set the controller, so only enter one number here.

Examples:

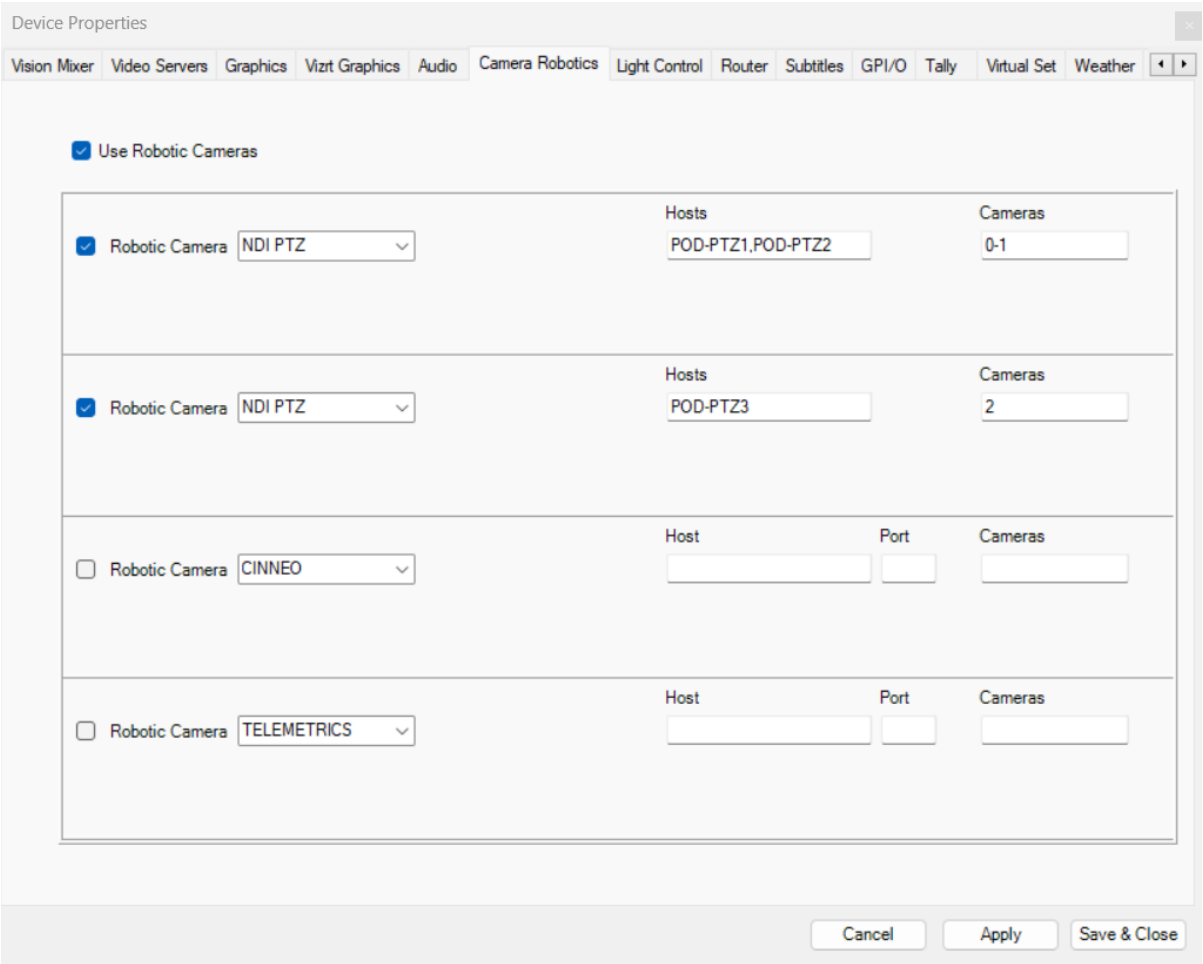
For NDI PTZ, you can configure multiple controllers each with one host:



or a controller with multiple hosts. This allows you to use the remaining controllers for other types of robotic cameras:



Or you can have combinations like following:



If multiple hosts are configured for a controller, the cumulative status will be shown in AV Automation and Mosart UI. If at least one host which is not in stand-by is not connected, the overall status will show disconnected (red). In the Standby menu in AV Automation however, you can see the status per device.

If, on the other hand, only a single host is configured for a controller (like the second controller in the figure above with Hosts *PD-PTZ3* and *Cameras 2*), the status of that host is shown in AV Automation and Mosart UI.

Camerobot Camera Systems

Viz Mosart can control anchor dependent shots for the Camerobot camera system. For details, please refer to Control Commands in Templates, section *Command Values and Parameters, DEVICE_PROPERTY*, using the value *CAMERA CONTROL*.

Camerobot can also be told how long a move should take, using the shot's Fade Time from the Template Editor. This is enabled with the `SendDuration` property, see Robotic Camera Configuration Files.

Robotic Camera Configuration Files

For both general and device-specific settings, see section Robotic Camera Configuration Files.

5.4.7 Light Control

The following systems are described:

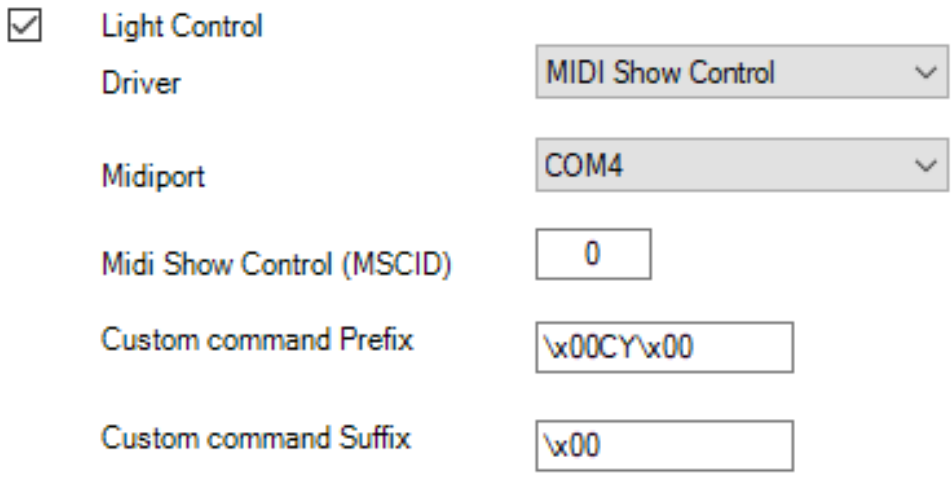
- Configuring Lighting Control
  - MIDI Show Control
  - OSC (Open Show Control)

Configuring Lighting Control

To configure lighting

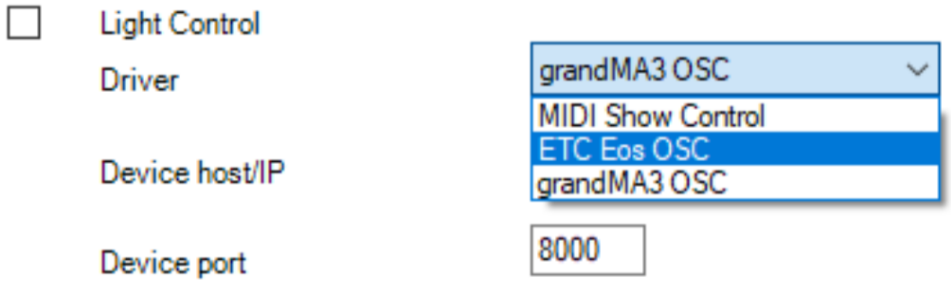
1. Navigate to **AV Automation > Devices > Properties > Light Control** tab.
2. Check the **Light Control** checkbox to enable one of the devices.

MIDI Show Control



- **Midiport:** System MIDI port to communicate with the light board.
- **Midi Show Control (MSCID):** The MIDI Show Control ID used for communicating with the light board.
  Default: *0*.
- **Custom command Prefix:** The prefix test sent with the MIDI Show Control command. Characters can be escaped using the x00 where 00 is replaced by the numeric value of the character.
  Default: *\x00CY\x00*.
- **Custom command Suffix:** The suffix sent with the MIDI Show Control command. Characters can be escaped using the x00, where 00 is replaced by the numeric value of the character.
  Default: *\x00*.

OSC (Open Show Control)



- **Driver**: either grandMA3 or ETC EoS
- **Device host/IP**: host name or IP address of the lighting console.
- **Device port**: port number specified on the lighting console for *receiving* OSC messages.

Viz Mosart supports OSC 1.1 (SLIP) via TCP.

grandMA3

In a simple setup, one OSC In&Out configuration should be created on grandMA3 that enables receiving commands from Viz Mosart.

For redundant Mosart Server setups, it is possible to create two OSC In&Out configurations on the device, matching IP addresses to main and backup.

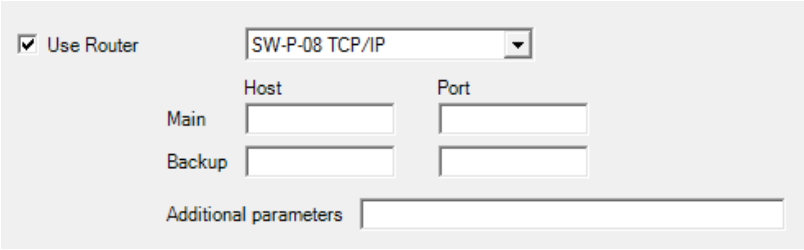
Viz Mosart's grandMA3 driver is implemented against the console's software version 2.3.

ETC Eos family

Enable OSC connection on the device under System/Show Control.

Viz Mosart's ETC driver is implemented against the console's software version 3.3.

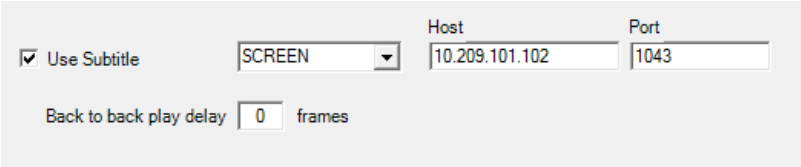
5.4.8 Router



- **Use Router:** Check to enable router control and select protocol from the list.
- **Router type:** EVERTZ QUARTZ, GVG G-SMS7000, SW-P-08 TCP/IP, SW-P-08 SERIAL, VENUS, VIKINX SERIAL or VIKINX TCP/IP.
- **Port:** Serial port (GVG G-SMS7000, Venus, SW-P-08 serial, and Vikinx serial).
- **Bit rate:** Selectable serial bit rate (GVG G-SMS7000).
- **Parity:** Parity bit, default None (GVG G-SMS7000).
- **Main Host:** Hostname or IP address of the main router (EVERTZ QUARTZ, SW-P-08 TCP/IP and Vikinx TCP/IP).
- **Main Port:** IP port of the main router (EVERTZ QUARTZ, SW-P-08 TCP/IP and Vikinx TCP/IP).
- **Backup Host:** Hostname or IP address of the backup router (SW-P-08 TCP/IP).
- **Backup Port:** IP port of the backup router (SW-P-08 TCP/IP).
- **Additional parameters:** For future use.

Copyright © 2026 Vizrt

5.4.9 Subtitles



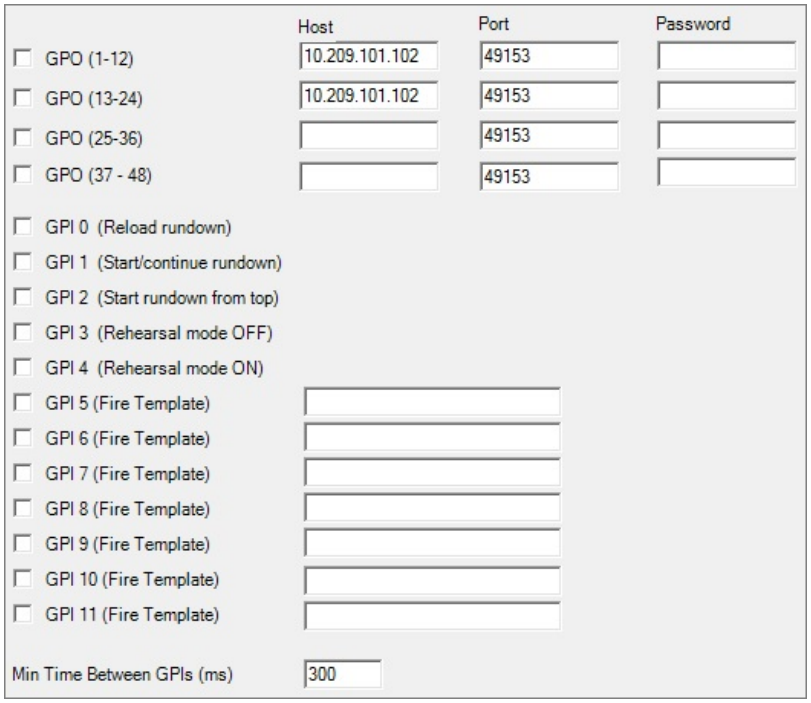
- **Use Subtitle:** Check to enable Screen Polystream subtitling.
- **Subtitling type:** SCREEN, SVT
- **Host:** Hostname or IP address of the subtitling system.
- **Port:** Defines the IP communication port.
- **Back to back play delay:** Delay in frames on the play command to the subtitling system when playing subtitle files back to back.

Copyright © 2026 Vizrt

5.4.10 GPI/O

Viz Mosart's GPI/O functionality supports sending out triggers to external equipment from a template (GPO) and also supports letting external equipment trigger templates from the currently active template set, as well as a selection of optional control commands (GPI). Viz Mosart can connect to GPI/O devices of the Web-IO product range from the vendor Wiesemann & Theis GmbH (W&T).

- GPO
- GPI
  - Triggering a Template
  - GPI Control Commands
- Troubleshooting
  - Verifying the Connection
  - Configuration Issues



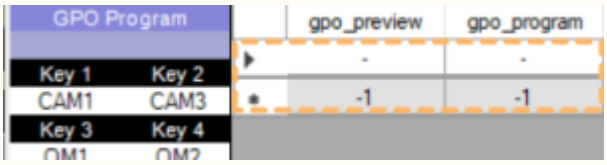
GPO

- **GPO (1-12) (check box):** Enables a Web-IO box for the first group of 12 GPOs. Note that this has to be configured for GPI 0-11 to be enabled.
- **GPO (13-24) (check box):** Enables a Web-IO box for the second group of 12 GPOs.
- **GPO (25-36) (check box):** Enables a Web-IO box for the third group of 12 GPOs.
- **GPO (37-48) (check box):** Enables a Web-IO box for the fourth group of 12 GPOs.
- **Host:** Hostname or IP address to the Web-IO GPI/O box.

Copyright © 2026 Vizrt

- **Port**: Port on the Web-IO GPI/O box.
  If configured as *80*, Mosart will use the HTTP Web-IO protocol, while for other Port values Mosart will use the Binary Web-IO protocol.
- **Password:** Defines the password, if needed.

Note: In Template Editor for the GPIO device, you enter *n* for *GPO n+1*. For example, with *0* for GPO *1* … *47* for GPO *48*.



GPI

Twelve GPI signals can be assigned in Viz Mosart. These GPI signals are received on the GPI/O device configured in the GPO (1-12), see above.

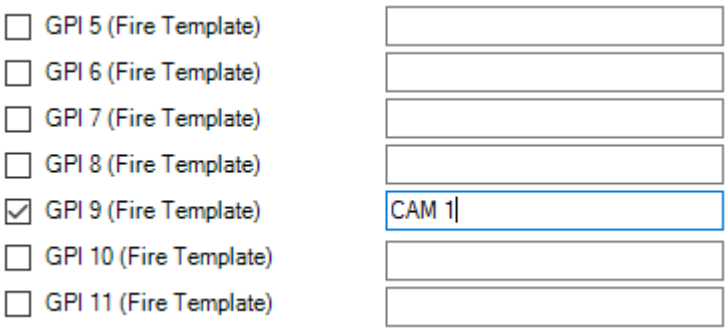
The first 5 GPI signals, GPI 0-4, are connected to fixed Viz Mosart commands. The GPI signals 5-11 can be configured to an arbitrary selection of operations.

- **GPI 0 (check box):** Reload rundown. The timeline is stopped and the current rundown is reloaded.
- **GPI 1 (check box):** Start/continue rundown. Will trigger the TAKE-NEXT event and with that start the timeline or advance to the next timeline element.
- **GPI 2 (check box):** Start rundown from top. Will start the rundown from the first story..
- **GPI 3 (check box):** Disable rehearsal mode.
- **GPI 4 (check box):** Enable rehearsal mode.
- **GPI 5-11 (check boxes/text boxes):** Enables firing of the template or command given in the text box - see below.
- **Min Time Between GPIs (ms)**: Threshold time before a new trigger command will be processed.

Each of the GPI signals 5-11 may be assigned to either trigger a Viz Mosart control command, trigger an arbitrary template from the active template set or trigger a direct take, as follows.

Triggering a Template

Taking a template using one of GPIs 5 to 11 (Fire Template) adds the corresponding template to the timeline.



To Trigger a Mosart Template

This can be used to trigger a Mosart template from the active template set:

Syntax: `[TemplateType]|[Variant]`

- Example: Camera|1

GPI Control Commands

As adding a template to the timeline (see *Triggering a Template* above) is not always wanted behavior, you can instead insert a *control command*.

To Trigger Mosart Control Commands

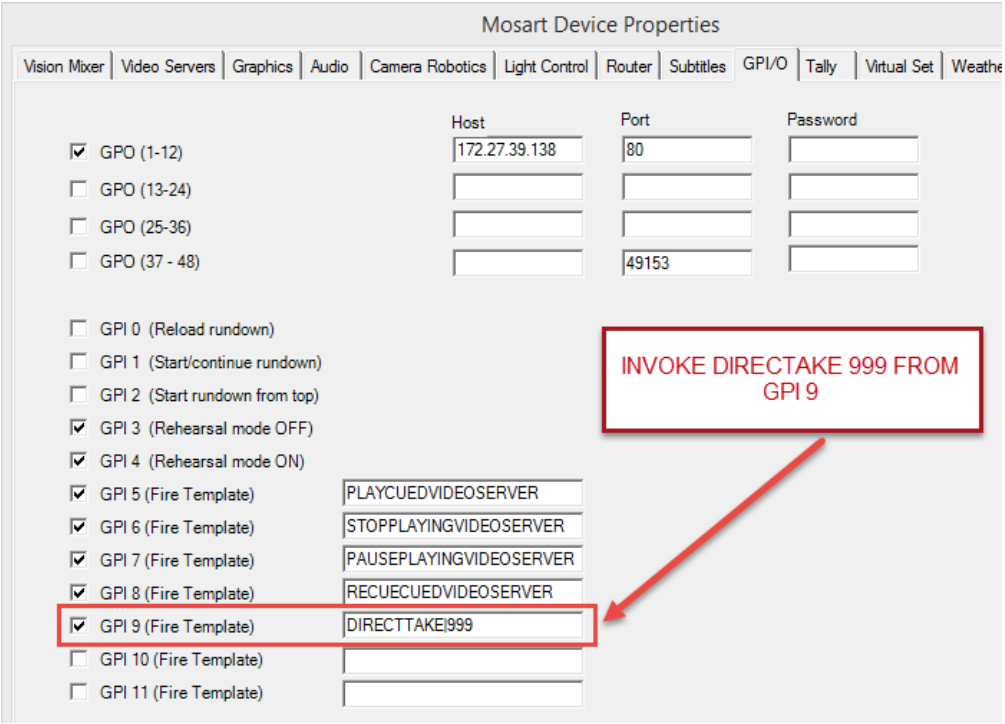
- To trigger a Mosart Control Command enter a valid command in the text field associated with the GPI signal. Supported commands are:
  - **DIRECTTAKE**: Trigger the numbered direct take.
  - **CLEARFOREGROUNDGRAPHICS**: Removes all CGs
  - **SKIPNEXTSTORY**: Skips next story **item**
  - **UNSKIPNEXTSTORY**: Un-skips next story **item**
  - **TAKELASTGRAPHICSOUT**: Removes last CG
  - **TAKEMANUALGRAPHICSOUT**: Removes manual CGs
  - **PLAYCUEDVIDEOSERVER**: Plays out the last cued clip on a video server
  - **PAUSEPLAYINGVIDEOSERVER**: Pauses the currently playing clip on a video server
  - **STOPPLAYINGVIDEOSERVER**: Stops the currently playing clip on a video server
  - **RECUECUEDVIDEOSERVER**: Recues the last cued clip on a video server

Direct Takes

A direct Take is an example of an available control command.

Syntax

DIRECTTAKE|<the Recall number of the direct take>
Example: DIRECTTAKE|999
will trigger direct take 999 (see screenshot below).



Troubleshooting

Verifying the Connection

Verifying

1. Assign a direct take to a GPI between 5 and 11.
2. Restart AV Automation.
3. Trigger this GPI from the external device connected to this GPI (see also To test GPI signals without using GPI below).
4. Verify on the Mosart server that the corresponding GPI is executed.
   Direct takes are shown in the AV Automation Log window (lower left) at Take.

To test GPI signals without using GPI

There is an simple method to test the GPI configuration.

- If a GPI/GPO device is configured, as shown above, *GPI signals 0-9* can be simulated inside AV Automation using keyboard sequence **CTRL+SHIFT+<number>** to trigger a corresponding GPI, where **<number>** is the GPI number *+1* (modulo 10).
  As examples, **CTRL+SHIFT+6** triggers GPI *5* and **CTRL+SHIFT+0** triggers GPI *9*.

Configuration Issues

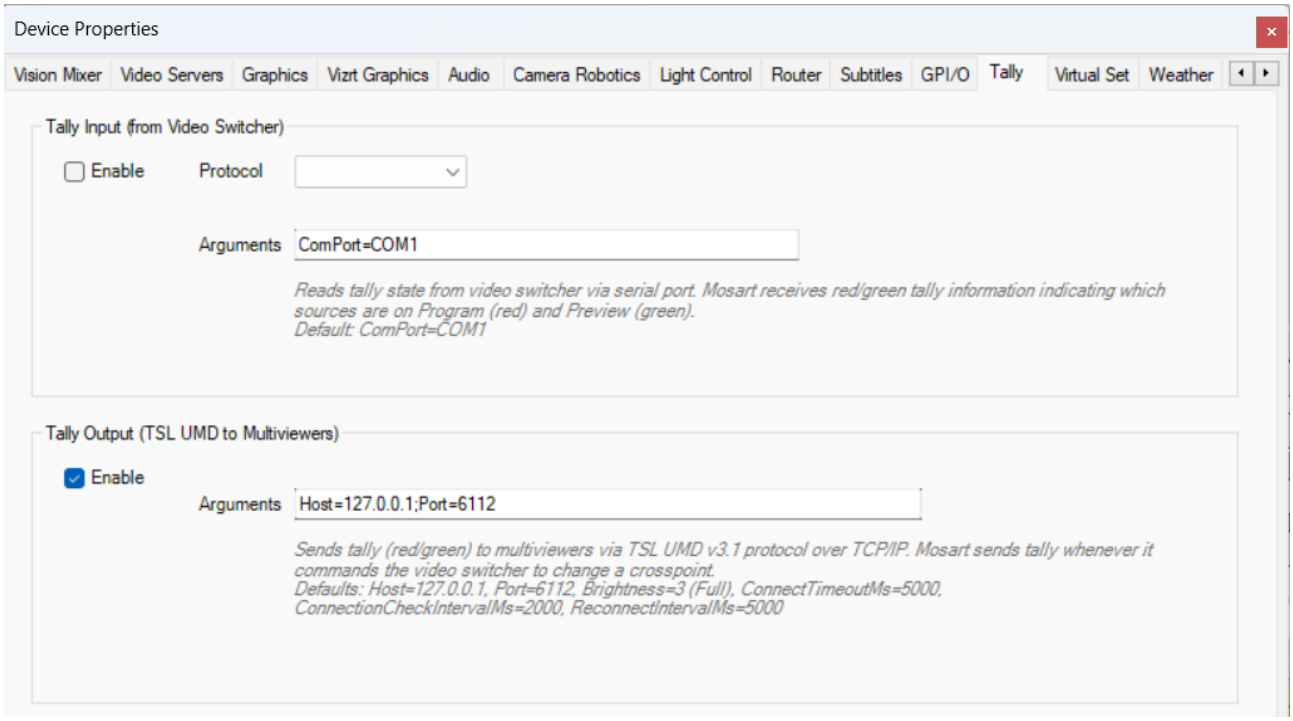
If the GPIO will not import the configuration file directly, a workaround is to save the current configuration and using Notepad++ or similar, to compare the difference between the new configuration and the existing

configuration.
Amend the configuration on the existing configuration to match the new settings. For an example, refer to the appendix GPIO with Sample XML Config.

5.4.11 Tally

The **Tally** tab in **Device Properties** has two independent directions. Each can be enabled on its own.

- **Tally Input**: Viz Mosart reads tally state from a hardware video switcher.
- **Tally Output**: Viz Mosart sends tally to multiviewers over TSL UMD v3.1.



Tally Input (from Video Switcher)

Reads tally state from a video switcher via serial port. Viz Mosart receives red/green tally information indicating which sources are on Program (red) and Preview (green).

- **Enable**: check to enable tally input.
- **Protocol**: GVG200 or SONY MVS.
- **Arguments**: `ComPort=COMx`. Default: `ComPort=COM1`.

Tally Output (TSL UMD to Multiviewers)

Sends tally (red/green) to multiviewers via the TSL UMD v3.1 protocol over TCP/IP. Viz Mosart sends tally whenever it commands the video switcher to change a crosspoint.

- **Enable**: check to enable tally output.

- **Arguments**: `Host=<ip>;Port=<port>`. Defaults: `Host=127.0.0.1`, `Port=6112`, `Brightness=3` (Full), `ConnectTimeoutMs=5000`, `ConnectionCheckIntervalMs=2000`, `ReconnectIntervalMs=5000`.

> **When tally output is most useful.** A traditional switcher reports its own tally, but a software switcher such as **Adaptive Live Producer** (Viz Engine driven through the Media Sequencer) produces none. There, Viz Mosart derives program and preview tally from the crosspoints it commands and sends it over TSL UMD. See the **Adaptive Live Producer: Tally** page for the rules specific to that workflow.

Protocol

TSL UMD v3.1 runs over TCP. Each packet is 18 bytes:

Byte	Content
0	Address (0 to 126).
1	Flags: Tally1 (bit 0), Tally2 (bit 1), Brightness (bits 4-5).
2-17	16-character label (source name from avconfig).

Tally1 (T1) = program (red). Tally2 (T2) = preview (green). Crosspoint numbers in `avconfig.xml` (`inputchannel` values) become TSL addresses. TSL UMD v3.1 supports addresses 0 to 126.
After a TCP reconnection, Viz Mosart resends the current tally state automatically. No manual action is needed.

TSL UMD Simulator

Mosart Test Suite includes a **TSL UMD v3.1 Simulator** (`TslUmdSimulator.exe`) that acts as a multiviewer receiver. Use it to verify tally addresses, states, and labels before connecting the real multiviewer.

Usage: `TslUmdSimulator [port] [logdir]`

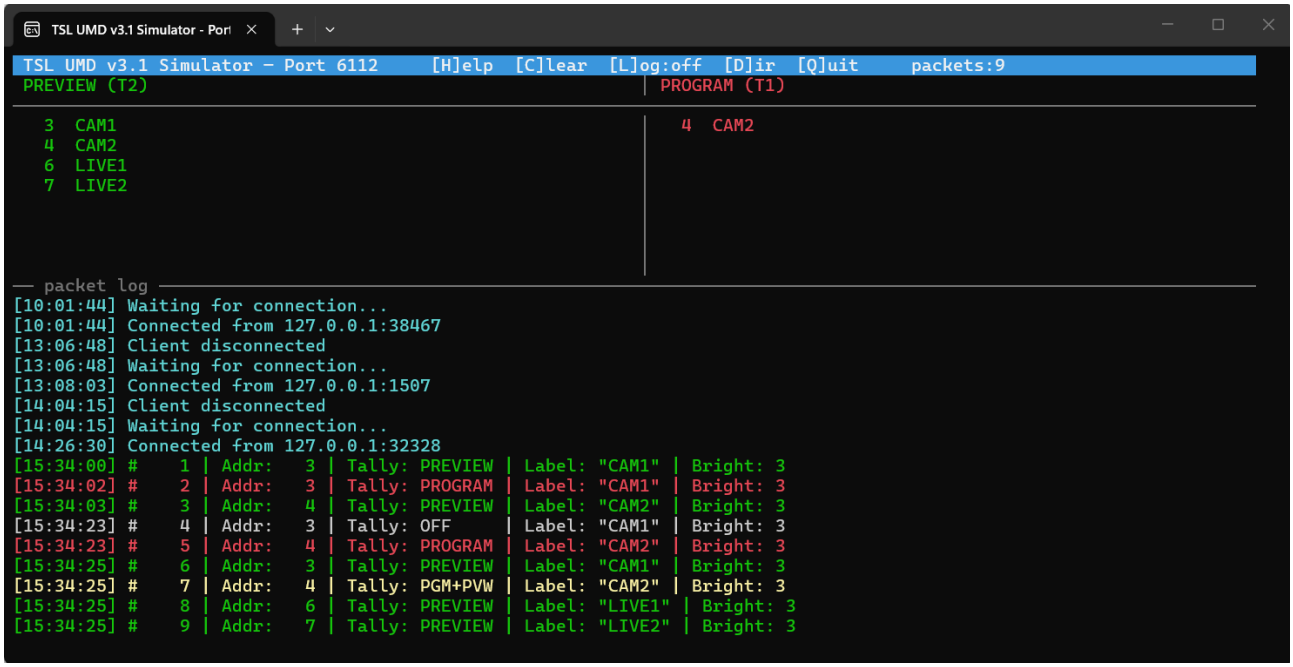
- `port`: TCP port to listen on (default: 6112).
- `logdir`: directory for log files (default: `C:\MMLogs`).

Commands during runtime:

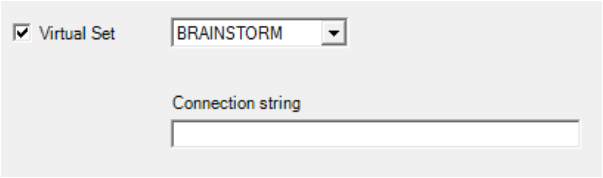
Key	Action
S	Show current tally summary.
L	Toggle logging to file. When enabled, every incoming tally event is written to `C:\MMLogs\TslTally_<timestamp>.log` in real time. Press L again to stop.
C	Clear console.

Key	Action
D	Change log directory.
Q	Quit.

The simulator shows each tally packet with color coding: red for program, green for preview, yellow for both, gray for off.



5.4.12 Virtual Set



- **Virtual Set:** Check to enable virtual set control.
- **Virtual Set:** BRAINSTORM, Viz Virtual Studio.
- **Connection string:** Server=hostname; [port=portnumber;] [mode=mixmode;].

Connection String Parameters for Viz Virtual Studio

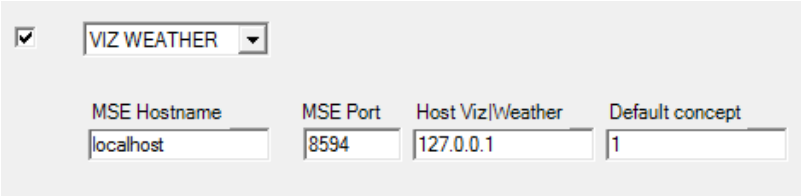
- **Server**: IP address or network name of machine where Viz Tracking Hub is running. Required.
- **Port**: TCP/IP port Viz Tracking Hub accepts connections. Optional.
  Default value is 20000.
- **LocalIP**: IP address of local network interface. Optional. Default value is first Ethernet interface.
- **Router**: Router name configured in Viz Tracking Hub. Optional.
  Default value is first router in Viz Tracking Hub.

Sample Connection string: *Server=10.45.0.15;Port=20000*

Camera set index enumeration starts from 1. This means that if a router has two camera sets configured, their indices will be 1 and 2.

Known Issue: Driver cannot detect when hub server is going offline. For example, if you kill the server, the driver keeps sending messages to server without an error being raised.

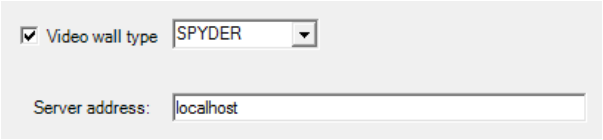
5.4.13 Weather



- **Use Weather:** Check to enable control of a weather control.
- **Weather controls:** VIZ WEATHER or WEATHER ONE
- **MSE Hostname (Viz Weather):** Hostname or IP address to the Media Sequencer controlling the Weather Control engine.
- **Hostname (Weather One):** Hostname or IP address of the Borealis Weather One engine.
- **MSE Port (Viz Weather):** Port to communicate with the Media Sequencer.
- **Port (Weather One):** Port to communicate with the Borealis Weather One engine.
- **Host Viz Weather (Viz Weather):** Hostname or IP address to the Viz Weather engine.
- **Default concept:** The weather concept to be used as default.

Copyright © 2026 Vizrt

5.4.14 Video Wall

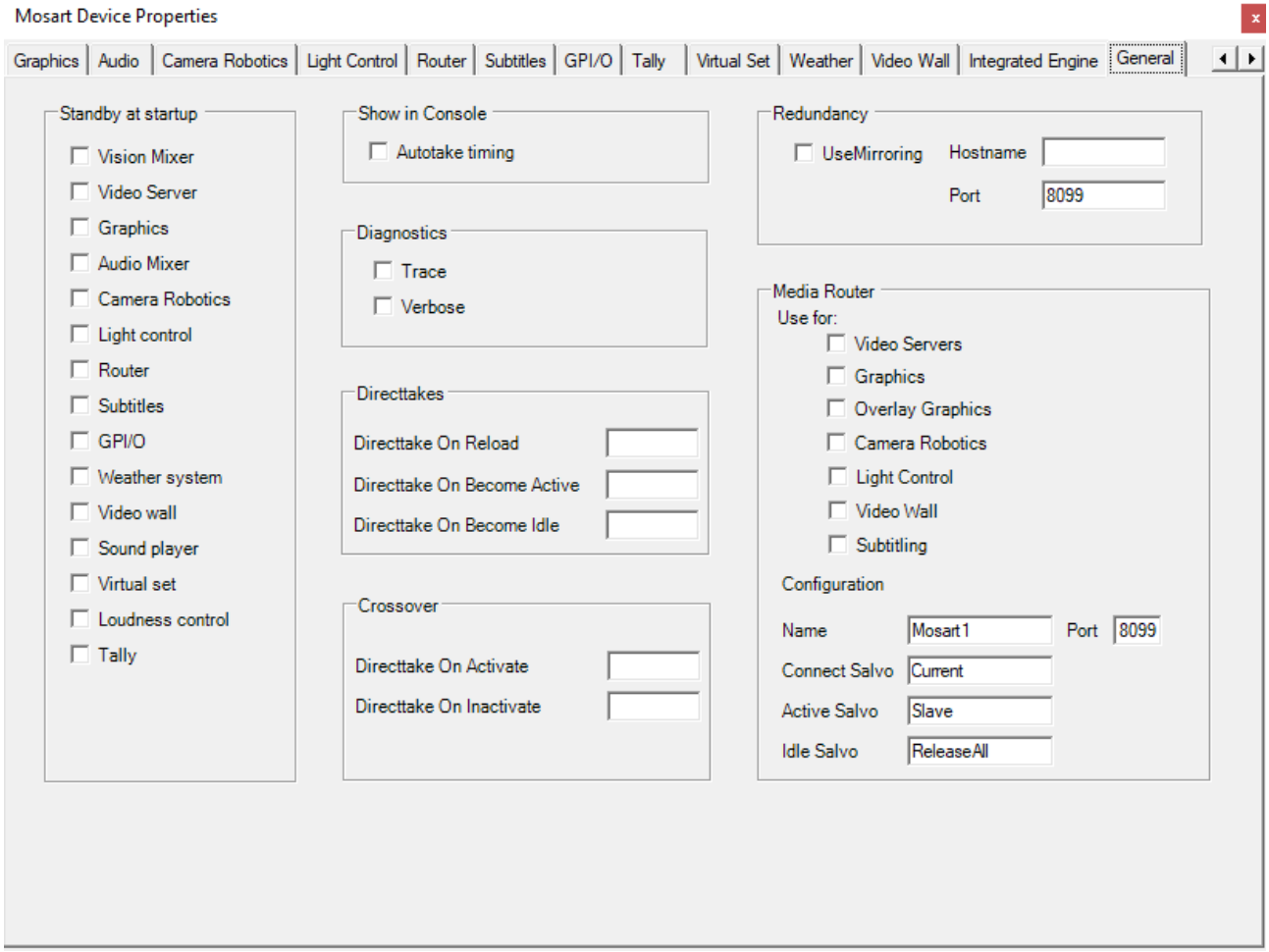


- **Video wall type:** Check to enable video wall control.
- **Video wall controls:** ENCORE, PANDORA, SPYDER, WATCHOUT (Version 5.2, DISPLAY CLUSTER PROTOCOL).
- **Port (Encore):** Serial port.
- **Bit rate (Encore):** Serial bit rate.
- **Server address (Pandora, Spyder, Watchout):** Hostname or IP address of the video wall server. For Pandora, the server is the PB Widget Designer.
  For WATCHOUT, full connection string. For details see Watchout Connection String.
- **Backup (Pandora):** Hostname or IP address of the backup PB Widget Designer.

See also

- Jupiter Systems PixelNet Video Wall Control

5.4.15 General



- **Standby at startup:** Selector for drivers that will start in standby mode. These selections must then be manually reset from standby mode, either using the Viz Mosart GUI or from AV Automation.

> **Note:** Whenever a Viz Mosart GUI is operational or if a backup Viz Mosart server is operational, they will *override* any standby settings made here. Standby settings applied in this menu only take affect at complete restart of all Viz Mosart applications. See the *Standby* description under **Main Menu** of the *Viz Mosart User Guide* for further details..

- Show in Console
  - **Autotake timing:** Adds timing to the log window. Used for debugging purposes.
- **Diagnostics**
  - **Trace:** Adds logging information to console used for dbugview.
  - **Verbose:** Increases logging information from AV Automation to the log file.
- **Directtakes**

    - **Directtake On Reload:** The Recall Nr (see Template Properties) of the direct take template to be fired upon re-loading the rundown.
    - **Directtake On Become Active:** The Recall Nr of the direct take template to be fired when the server becomes active from idle, when switching between main and backup Viz Mosart servers.
    - **Directtake On Become Idle:** The Recall Nr of the direct take template to be fired when the server becomes idle from active, when switching between main and backup Viz Mosart servers.
- **Crossover**
    - **Directtake on Activate:** The Recall Nr of the direct take template to be taken when this server goes active in crossover switch.
    - **Directtake on Inactivate:** The Recall Nr of the direct take template to be taken when this server goes inactive in crossover switch.
- **Redundancy**
    - **UseMirroring:** Check to enable mirroring. Upon enabling mirroring on both main and backup server, the server states are synchronized.
    - **Hostname:** Hostname or IP address of the Backup Server if using Main Server, and vice versa.
    - **Port:** IP port number of the Backup Server if using Main Server, and vice versa.

> **Note**: If Redundancy is to be used, the **Media Router** Port has to also be set, and then to the same value as the **Redundancy** Port. Recommended value: 8099.

- **Media Router**
    - **Use for:** Select which controls should be used by the Media Router. (Also see the MMR Tab for Overlay Graphics Interface configuration).
    - **Name:** Media router's hostname or IP address.
    - **Port:** Media router's connection port number. Must be the same as **Redundancy** Port. Recommended value: *8099*.
    - **Connect Salvo:** The salvo name to be taken when this server connects after restart. Default: Current.
    - **Active Salvo:** The salvo name to be taken when this server becomes the active server from being idle. Default: Slave.
    - **Idle Salvo:** The salvo name to be taken when this server becomes idle after being active. Default: ReleaseAll.

Copyright © 2026 Vizrt

5.5 A/V Setup (Audio and Video Setup Config File)

5.5.1 Working with Audio and Video Setup

In the AV Automation menu **Audio and Video Setup**, you can add audio and video channels and configure their attributes.

Viz Mosart relies on predefined *video crosspoints* entered into AV Automation at initial installation and setup of Viz Mosart.

- XML Editor Utility
- Video Configurations
- Audio Configurations
- Template-driven Effects
- Router Source Configurations
- Router Destination Configurations
- Engine Presets
- Engine Top Scenes

XML Editor Utility

A simple XML editor utility (A/V Setup editor) is used to manage and configure the settings in AV Automation.

> ⚠ **Note**: All *comments* in an existing avconfig.xml are removed after editing with the A/V Setup editor.

To open the avconfig utility

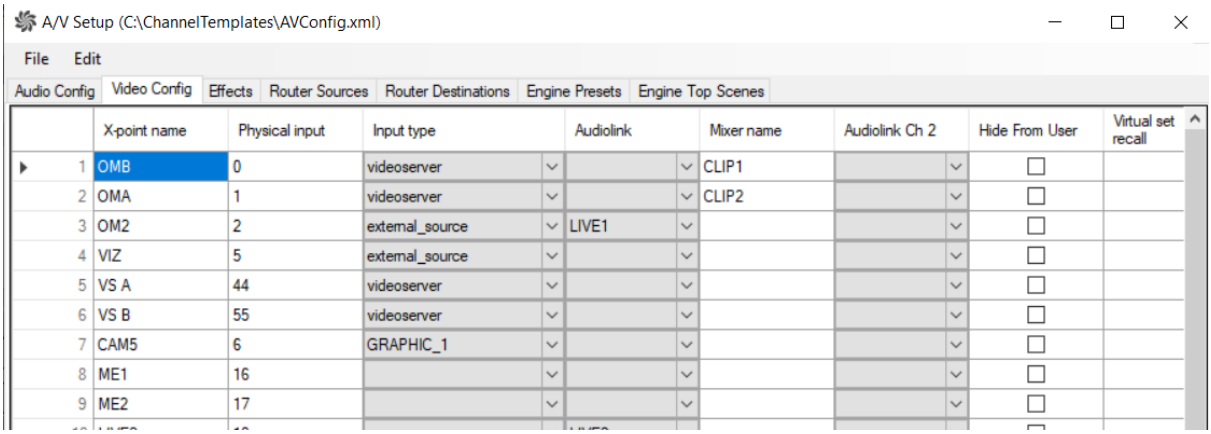
- From AV Automation, select **Devices > A/V Setup**.

This presents *all* configuration values. You can filter a specific set of configurations, for example *video* configurations, as explained below.

Video Configurations

To work with video crosspoint configurations

1. From AV Automation, select **Devices > A/V Setup > Video Config** tab.



	X-point name	Physical input	Input type	Audiolink	Mixer name	Audiolink Ch 2	Hide From User	Virtual set recall
1	OMB	0	videoserver		CLIP1		☐	
2	OMA	1	videoserver		CLIP2		☐	
3	OM2	2	external_source	LIVE1			☐	
4	VIZ	5	external_source				☐	
5	VS A	44	videoserver				☐	
6	VS B	55	videoserver				☐	
7	CAM5	6	GRAPHIC_1				☐	
8	ME1	16					☐	
9	ME2	17					☐	

2. As a minimum, you should enter the crosspoint details: **X-point name**, **Physical input** and **Input type** (where applicable).

Settings

- **X-point name:** Name to use for the video crosspoint in the Viz Mosart system. This name will be displayed in the **Crosspoints** drop-down menu in all **Templates**, in the GUI drop down menus (newsroomtags, etc.) and in the **NRCS Plugin**.
  Notes:
  - **Must be configured in A/V Setup**
    Any crosspoint name used in a template must exist in A/V Setup.
    Example: If you reference “CAM1” in a template, ensure that “CAM1” is configured as an X-point name in A/V Setup. Otherwise, the command will silently fail and not switch the source.
  - **Spaces are stripped**
    Mosart removes all spaces from crosspoint names internally.
    Example: If you configure an X-point name “CAM 1,” Mosart treats it as “CAM1” internally. Referencing “CAM 1” in a template will still match, because spaces are ignored at lookup time.
    Recommendation: For clarity, avoid using spaces in crosspoint names (e.g., use “ME1” instead of “ME 1”).
  - **Case Insensitivity**
    Mosart also ignores case when matching crosspoint names.
    Example: “cam1,” “CAM 1,” and “CAM1” all resolve to the same crosspoint in Mosart.
- **Physical input:** Input on the video switcher. These are the physical inputs of the video switcher.
- **Audiolink:** Audio fader to link to the video source when the video source is changed in the Newsroom Control System or Viz Mosart GUI, enabling the audio faders to follow the video crosspoints.
  The drop-down menu presents Audiolink-values corresponding to the audio sources that may be selected for a fader, alongside the selected input source, for example a camera.
- **Audiolink Ch 2:** Second audio fader link for the video source. The fader name that will be used when enabling video server Ch2 in the Template Editor. When the video source is changed in the Newsroom Control System or Viz Mosart GUI, the audio faders will follow video crosspoints.
  Should be used for audio mixers with mono-faders which cannot be coupled in the mixer.
- **Mixer name**: The name of the crosspoint as it is known *on the switcher itself*, for cases where that differs from the **X-point name** used inside Viz Mosart. Leave empty when the switcher recognizes the **X-point name**

directly.

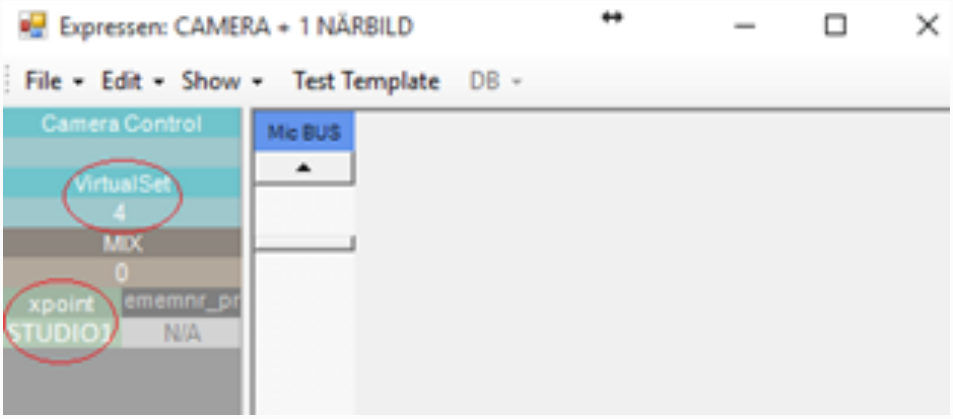
Originally introduced for the legacy Viz Opus, this setting is also used by current drivers:

- **Viz Vectar Plus / TriCaster (NDI router):** *Required*. Must hold the hardcoded switcher-side name (typically INPUTn, DDRn, BFRn, MIX n). See Template Device Functions.
- **Panasonic Kairos:** *Required*. Used as the source name in all background bus, AUX and key-fill commands.
- **Viz Engine as a Switcher:** Used when the **X-point name** is a Viz Mosart user-friendly name rather than the engine's source ID. See Viz Engine as a Switcher.

Note: Behavior when **Mixer name** is empty depends on the driver. Viz Engine falls back to the **X-point name**; TriCaster/Vectar falls back to the **Physical input** number; Panasonic Kairos has no fallback, so the value must be set.

- **Input type:**
  - **camera:** Select to assign this crosspoint as a camera type. This is used by the **Audio Panel** application.
  - **external_source:** Select to assign this crosspoint as an external source. This is used by the **Audio Panel** application.
  - **videoserver:** Select to assign this crosspoint as a server channel.
  - **GRAPHIC_1-5:** Select to assign this crosspoint as a graphics engine.
  - **ROUTER_RIPPLE_A:** Select to use as router source *A*.
  - **ROUTER_RIPPLE_B:** Select to use as router source *B*.
- **Hide from user:** Do not show crosspoint to user. This prevents the crosspoint from displaying in the Viz Mosart GUI drop-down menus (Newsroom tags etc.) and in the NRCS Plugin. Default: *Unchecked.*
- **Virtual set recall:** Value of a video cross point that when called, will *also recall* a virtual set, effectively binding a cross-point to a virtual studio setting.

Info: A virtual set recall can also be performed from a template.

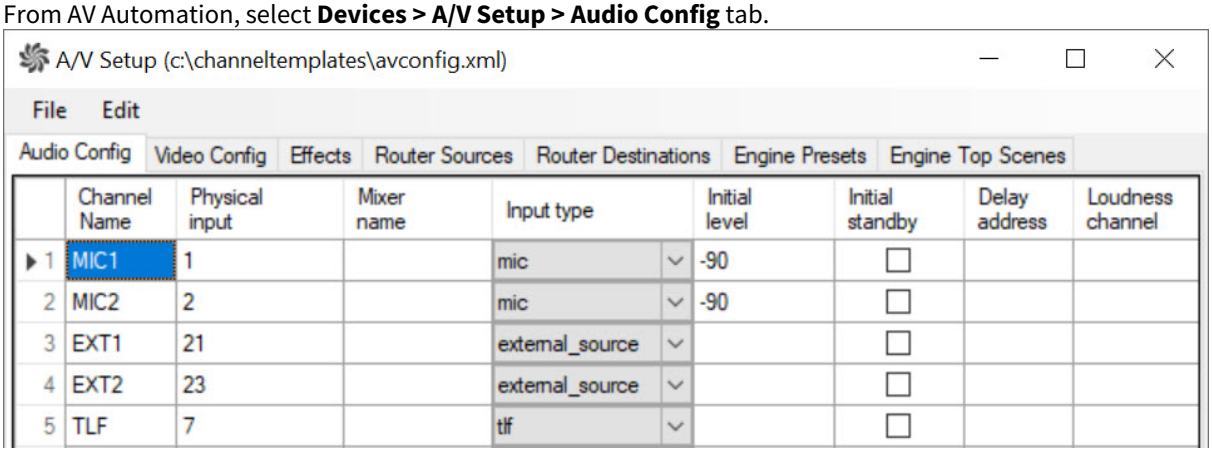


- **Comments:** Only for useful setup notes, in this menu. Not visible or used anywhere else in the application.

Audio Configurations

To work with audio mixer configurations

1. From AV Automation, select **Devices > A/V Setup > Audio Config** tab.



2. As a minimum, you should enter **Channel Name** and **Physical input**.

Settings

- **Channel Name:** Name to use for the audio fader in the Viz Mosart system (for referencing the audio channel in Viz Mosart templates).
- **Physical input:** For audio mixers that use fader numbers or MIDI assigned faders, this number must correspond with the fader number on the audio desk. Fill it in with a number that is unique within the column even when your desk is addressed by name — several drivers use it as Viz Mosart's internal key for the channel, and a channel whose **Physical input** is empty or non-numeric is skipped at start-up, after which level, mute and PFL have no effect on it.
  Two exceptions, where filling it in is harmless but not required: The **Studer Vista** driver silently replaces a blank or duplicate number with one of its own, and drivers that learn the fader assignment from the desk (**Stagetec Diamond RAS**) ignore the value entirely.
- **Mixer name:** A unique fader identifier, for audio desks that use fader names instead of numbers.

Info: For LAWO EMBER+, these types of faders are controllable and may be configured:
- Auxes, groups, inputs, sums, vcas.
- Use the value of the *startup* label, found in the Ember+ tree/viewer, located under mixers/primary/channels for the respective fader types.
  (vcas are instead found under mixers/global/vcas).
- For a *stereo* fader, use the *left*. *Do not* configure the *right*.

Info: For STAGETEC EMBER+, the controllable channels are the ones under the CXP / channel node in the Ember+ tree. Use the fader Identifier (for example *c#1*).

Info: For TASCAM SONICVIEW 16 EMBER+, the controllable channels are the ones under the sonicview/channels node in the Ember+ tree. Use the fader Identifier (for example *channel_1*).

Info: For LAWO PowerCore, the controllable channels are the ones under the PowerCore/Sources and PowerCore/Sums nodes in the Ember+ tree. (If the Identifier of the first level node is different from 'PowerCore', set that Identifier in the FirstLevel config property in AudioMixerEmberPlus.xml.) Use the node Identifier (for example *MPX01*).

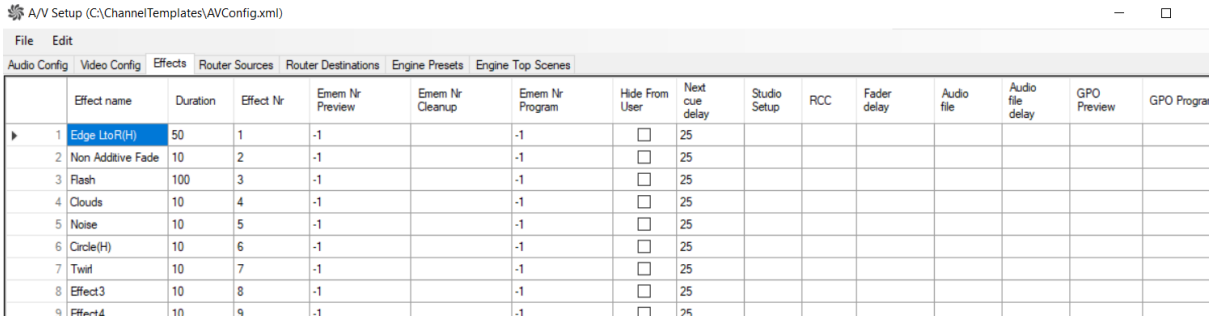
- **Input type:** For the internal Viz Mosart audio logic, the input type of the fader must be set. This can also be done in the **Template Editor**.
- **Initial level:** Desired level for the audio fader to obtain after a reload of the Viz Mosart rundown.
- **Initial standby:** Selects the faders that should *not* be reset at the start-up of the **AV Automation** application. Default: *Unchecked.*
- **Delay address:** (Lawo non-Ember+ audio mixer boards only) Gives the signal address for the delay function.
- **Loudness channel:** Defines the channel number for loudness level control.

Template-driven Effects

You can define effects in Viz Mosart that can then be recalled by a template. This avoids having to recreate the effect in each template that refers to it.
Use the **Effects** table, which is then referenced during the execution of a template that has specified an **Effect name** or **Effect Nr**.

To manage vision mixer effects configurations

1. From AV Automation, select **Devices > A/V Setup > Effects** tab.



	Effect name	Duration	Effect Nr	Emem Nr Preview	Emem Nr Cleanup	Emem Nr Program	Hide From User	Next cue delay	Studio Setup	RCC	Fader delay	Audio file	Audio file delay	GPO Preview	GPO Progra
1	Edge LtoR(H)	50	1	-1		-1	☐	25							
2	Non Additive Fade	10	2	-1		-1	☐	25							
3	Flash	100	3	-1		-1	☐	25							
4	Clouds	10	4	-1		-1	☐	25							
5	Noise	10	5	-1		-1	☐	25							
6	Circle(H)	10	6	-1		-1	☐	25							
7	Twirl	10	7	-1		-1	☐	25							
8	Effect3	10	8	-1		-1	☐	25							
9	Effect4	10	9	-1		-1	☐	25							

2. An *effect* may perform various actions on the vision mixer, the RCC, the Audio player, the GPO, and possibly other devices.

Note: When using an entry in the predefined Viz Mosart **Effect** table, the EMEM *must include the transition* between bus A and B, since transition is not performed by Viz Mosart.

Settings

- **Effect name:** Name or description of the effect. *Mandatory*.
- **Effect Nr:** Value of the effect assigned from the newsroom system. Used in conjunction with **Studio Setup** to uniquely identify effects. *Mandatory*.

- **EMEM Nr Preview:** Video switcher preset to recall when the template is taken in the preview. *Mandatory*.
- **EMEM Nr Program:** Video mixer preset to recall when the template is taken on the program. *Mandatory*.
- **Hide From User:** There are cases where you have an effect that should only be run under specific circumstances (for example, on an ME row), this effect being recalled from specific templates. If such an effect is unintentionally selected from the Viz Mosart GUI (or NRCS plugin) this could have unintended on-air consequences.
  To prevent this, *Hide From User* can prevent the effect displaying in the Viz Mosart GUI **Transition** panel and in the NRCS Plugin. (requires Mosart web apps version 2.5) Default: *Unchecked.*
- **EMEM Nr Cleanup:** This register will be recalled when cueing the next template. Use only if the register used for the effect demands another "cleanup" register recall to prepare the mixer for the next events.
- **Duration:** Duration of the effect in frames. Will hold the triggering of the template following the effect, for this amount of time.
- **RCC:** Recall a **Robotic Camera Control** shot.
- **Next cue delay:** Used to hold the cueing in the preview of the second template/element after the effect. The number of frames entered will be in addition to what is already set as Min. cue delay in the **Video Switcher** setup.

Info: Next cue delay is ignored if **EMEM Nr Program** has a negative value!

- **Fader delay:** Delay opening the audio effect fader (the **Audio Player** fader). Value is given in frames. Ignored if this value is greater than **Next cue delay**.
- **Audio file:** Viz Mosart can play an audio file on **Audio Player** synchronous with the mixer effect. Specify the filename for the audio file here.
- **GPO Preview:** Send GPO # when entering Preview mode.
- **GPO Program:** Send GPO # when entering Program mode.
- **Macro Preview:** Refer to the **Text** field in **Recall preview** in the Template Device Functions - Macro Recall section.
- **Directtake:** Number of Direct Take template triggered when performing an effect transition. Often used to allow control of additional devices.
  For example, this could be a direct take triggering a named overlay graphic or a direct take with additional commands required by the vision mixer for this effect.
- **Studio Setup:** Specifies a specific studio setup for which the effect will be run. This attribute is optional and is used in conjunction with **Effect Nr** to uniquely identify effects.
  See Effect number and Studio Setup below.
- **Macro Preview Action:** Refer to the **Drop-down** description in **Recall preview** in the Template Device Functions - Macro Recall section.
- **Macro Program Action:** Refer to the **Drop-down** description in **Recall program** in the Template Device Functions - Macro Recall section**.**

Note: Macro Preview Action, **Macro Program Action**
Load Project could affect the performance of the switcher and should thus be used with caution during a running show.

- **Macro Program:** Refer to the **Text field** details in **Recall program** in the Template Device Functions - Macro Recall section.

Effect Number and Studio Setup

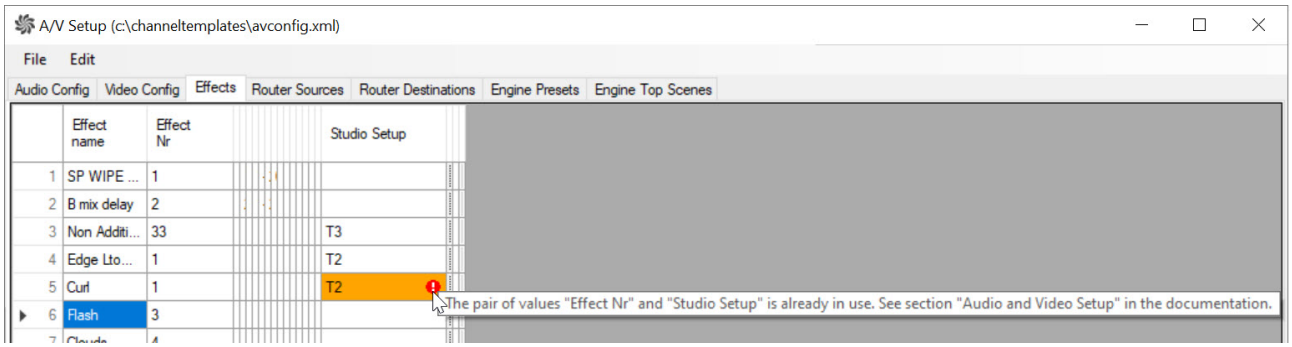
Normally the effects are shared between all the studios/control rooms. But in some cases, a given effect is required to function differently in one studio setup than another. This is achieved by adding an extra entry for that effect, but with the **Studio Setup** that this effect is reserved for. The rules for how this works are as follows:

- The combination of **Effect Nr** and **Studio Setup** must be unique within the **Effects** table.
  Multiple effects can have the same **Effect Nr**, but these must have *different* values in **Studio Setup**.
- An effect which has **Studio Setup** specified will only be executed for the corresponding studio setup.
- An effect without **Studio Setup** specified will be executed for *all* studio setups that do not have any dedicated effect specified.

Viz Mosart processes an effect in the following order:

- Search for an effect with given **Effect Nr** and with **Studio Setup** that matches the *active* Studio Setup.
- If no match is found, search for an effect with given **Effect Nr** and *empty* **Studio Setup**.

If there are duplicate values of **Effect Nr** and **Studio Setup** the cell will be highlighted and an error message will indicate the reason for the warning.

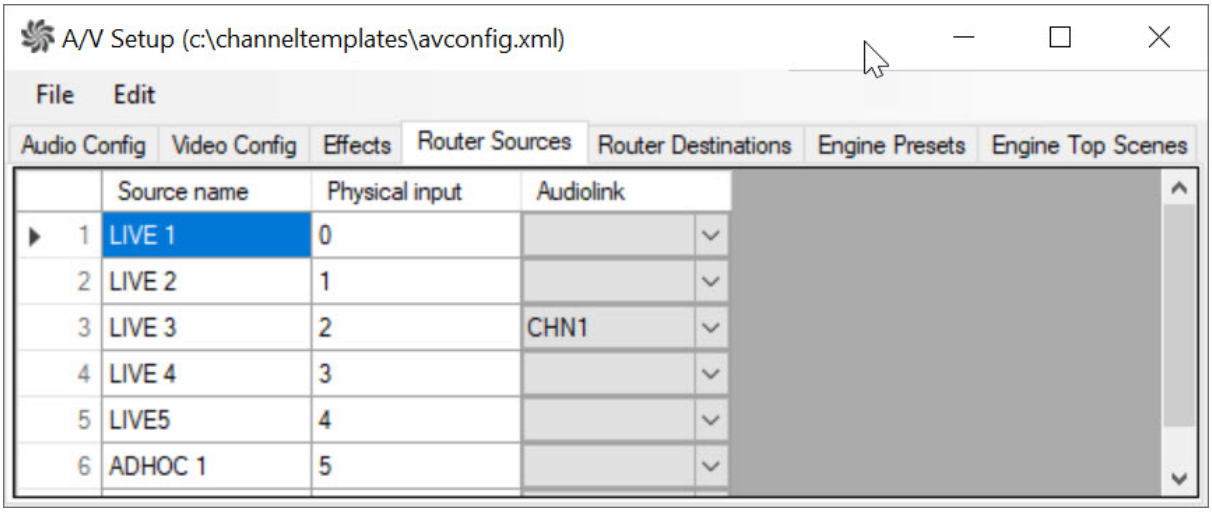


Router Source Configurations

Router source crosspoints can act as an emergency switcher if the vision mixer fails to switch. Alternatively, a router switch can change a background wall source in the studio, or anything in your production that requires switching.

To manage router source configurations

1. From AV Automation, select **Devices > A/V Setup > Router sources** tab.



2. Router switches are predefined as part of a Viz Mosart template. Keyboard shortcuts can also be programmed for a router switch to take place.

Settings

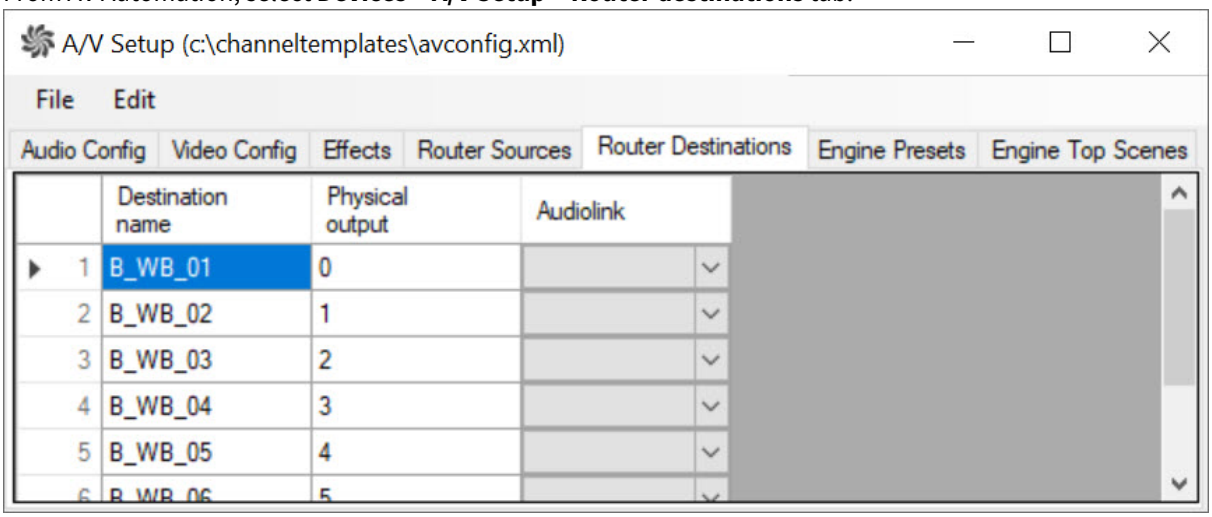
- **Source name:** Name to use for the router source in the Viz Mosart system.
- **Physical input:** Physical or virtual input to the router.

Router Destination Configurations

Router destination crosspoints can act as an emergency studio output if the vision mixer fails to switch. Alternatively, a router destination could be a monitor in the studio, a transmission output, or anything that needs switching as part of your production.

To manage router destination configurations

- From AV Automation, select **Devices > A/V Setup > Router destinations** tab.



Settings

- **Destination name:** Name to use for the router destination in the Viz Mosart system. This name will be displayed in the router crosspoints drop-down menu in all templates and in the ActiveX application.
- **Physical output:** Physical or virtual output on the router.
- **Audiolink**: Linked device.

Engine Presets

- This tab contains settings needed when using Viz Engine as a vision mixer.
  Refer to the *Presets* topic of Viz Engine as a Switcher.

Engine Top Scenes

- This tab contains settings needed when using Viz Engine as a vision mixer.
  See *Top Scenes* section of Viz Engine as a Switcher.

Copyright © 2026 Vizrt

5.6 Device Connection Strings and Configuration Files

This section contains an overview of Device Connection Strings and Configuration Files for connecting to equipment in your broadcast environment.

A *device* in this context is a physical unit, such as an Viz Engine or a video server. The *device driver* is the code that provides Viz Mosart the ability to handle and control the actual device. The *Device Connection string* is a textual description that identify the type, name and various other parameters required for Viz Mosart control the device. In some cases, you need to provide additional parameters for the device, typically specified in an XML file, in addition to the Connection String.

This section contains the following topics:

- Video Wall Connection Strings
- Device Configuration Files
- Video Server, MAM and Database Connection Strings

Copyright © 2026 Vizrt

5.6.1 Video Wall Connection Strings

Video wall connection strings are used by AV Automation. This section contains Connection String types.

> **Info:** For general information on connection strings, see Video Server, MAM and Database Connection Strings.

Watchout Connection String

This connection string is used for WATCHOUT video walls.

Modes of Operation

The driver may operate in any of three different *modes*. (For the sake of simplicity, assume that at least two commands are to be performed in one operation).

- **default:** A driver operation sends the commands one after the other. It waits for WATCHOUT to respond finally (or time-out) before sending the next command. If a command fails, the rest is skipped. The advantage is that when the driver operation finishes, the entire operation is complete (or has failed). However, as driver calls may arrive in different threads, and therefore simultaneously or overlapping, problems may occur. (The driver is made for performing (sending and feedback-waiting) one command at a time.)
- **SequenceRecallAndMixes:** A driver operation *queues* the commands to be performed as a unit (without interference from other such units), possibly at a later time. The problem mentioned above is solved, however at the cost of the driver operation possibly finishing before the commands are complete.
- **SequenceIndividualCommands:** A driver operation *queues* the *first* command (to be performed without interference from other commands). If successful, the *next* command is queued, and so on. This *may* be useful *if* WATCHOUT is able to keep several shows loaded at the same time (and is able to use the run timeline argument to choose between them).

Examples

AV Automation Connection String:

```
AuthenticateTimeout=[authenticateTimeout];BufferSize=[bufferSize];BusyNonCommandTimeo
ut=[busyNonCommandTimeout];BusyTimeout=[busyTimeout];GetStatusThreshold=[getStatusThr
eshold];LoadTimeout=[loadTimeout];PingInterval=[pingInterval];PingTimeout=[pingTimeou
t];Port=[port];ReconnectInterval=[reconnectInterval];ResetAnyway=[resetAnyway];ResetT
imeout=[resetTimeout];RunTimeout=[runTimeout];SequenceIndividualCommands=[sequenceInd
ividualCommands];SequenceRecallAndMixes=[sequenceRecallAndMixes];Server=[server];UseG
etStatus=[useGetStatus]
```

Example:

```
Server=192.168.97.129;AuthenticateTimeout=200;RunTimeout=40
```

Connection String Properties - Watchout

String	Type	Description	Default
AuthenticateTimeout	integer>0	The time (in ms) allowed for WATCHOUT to process an authenticate command and respond with Ready.	`2000`
BufferSize	integer>0	The maximum number of bytes to receive from WATCHOUT at a time. Should be set at least as large as the longest conceivable WATCHOUT response.	`1024`
BusyNonCommandTimeout	integer>0	The time (in ms) allowed for WATCHOUT to return to normal operation after a non-command Busy.	`10000`
BusyTimeout	integer>0	The time (in ms) allowed for WATCHOUT to increase the % of otherwise equivalent Busy responses.	`60000`
GetStatusThreshold	integer	The highest acceptable value of the third parameter of the Reply feedback to the getStatus command General health status of the cluster: • 0: OK • 1: Suboptimal • 2: Problems • 3: Dead	`1` (Suboptimal)
LoadTimeout	integer>0	The time (in ms) allowed for WATCHOUT to process a load command and respond with Ready.	`10000`
PingInterval	integer>0	The time (in ms) between ping commands sent to WATCHOUT.	`10000`
PingTimeout	integer>0	The time (in ms) allowed for WATCHOUT to process a ping command and respond with Ready.	`10000`
Port	integer>0	The port to connect to.	`3039`
ResetAnyway	true/false	Whether reset commands are always sent before run commands even if not present in the RecallAndMix argument.	false
ResetTimeout	integer>0	The time (in ms) allowed for WATCHOUT to process a reset command and respond with Ready.	`10000`

Copyright © 2026 Vizrt

String	Type	Description	Default
RunTimeout	integer>0	The time (in ms) allowed for WATCHOUT to process a run command and respond with Ready.	`10000`
ReconnectInterval	integer>0	The amount of time (in ms) to wait to reconnect after a failed connect or authenticate.	`10000`
SequenceIndividualCommands	true/false	Whether the driver should operate in that mode. Has no effect when SequenceRecallAndMixes=true.	false
SequenceRecallAndMixes	true/false	Whether the driver should operate in that mode.	false
Server	string	The DNS name of WATCHOUT.	localhost
UseGetStatus	true/false	Whether the Ping operation should use the getStatus command instead of the ping command.	false

5.6.2 Device Configuration Files

Location of Device Configuration Files

The device configuration files are located in the following directory: *%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\ConfigurationFiles.*

Note:

- Do not modify any files at this location, as they are overwritten by the next installation (update/upgrade).
- Copy any files that must be modified to the " `%ProgramData%\Mosart Medialab\ConfigurationFiles"` directory, and modify them there.
  This ensures that configurations remain valid after system upgrades or downgrades.
- Historically, `C:\channeltemplates` has been the folder to copy to, but with a few exceptions, Viz Mosart applications will look in `%ProgramData%` before checking `C:\channeltemplates` for configuration files.

Name of the configuration file(s)
In the following sections, the configuration files are shown with their *default* names. For some device types it is possible to specify a different configuration file name in the device connection string.
This makes it possible to use different configurations for various devices of the same type, for example when you have two slightly different settings for two video servers of the same type and brand.
There *must* be a 1:1 match between the name of the *configuration file* and the name referred to in the device *connection string*.

This section contains the following topics:

- Audio Mixer Configuration Files
- Engine Switcher Configuration File
- Graphics Configuration Files
- Light Control Configuration Files
- Robotic Camera Configuration Files
- Subtitling Configuration Files
- Video Router Configuration Files
- Video Server Configuration Files
- Vision Mixer Configuration Files

Information about *Video Server* Configuration Files is in section, Video Server, MAM and Database Connection Strings.

Audio Mixer Configuration Files

These config files are initially located in the Viz Mosart Server installation folder in a sub folder, *Configuration Files*. For operations, you copy the file to "%ProgramData%/

- Ardour Configuration File
- Calrec TCP/IP Configuration File
- Lawo Ember+ Configuration File
- LAWO PowerCore Configuration File
- MidiMixer Configuration File
- Stagetec Configuration File
- Stagetec Ember+ Configuration File
- Studer 3000 TCP/IP Configuration File
- Studer Vista Configuration File
- Tascam Sonicview 16 Ember+
- General Audio Mixer Configuration File

- See also Audio.

Ardour Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Ardour configuration file named *AudioMixerArdour.xml*.

Please see these files for the available settings and their documentation.

Calrec TCP/IP Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Calrec TCP/IP configuration file named *AudioMixerCalrec.xml*.

Please see these files for the available settings and their documentation.

Info: AckTime and ConWait
The sample *AudioMixerCalrec.xml* file includes two settings *AckTime* and *ConWait* with their respective default values.

- If you need to change the value of these defaults, you must at the same time, change the naming to all *lowercase* (*acktime* and *conwait*) for the change to be accepted and applied.

Lawo Ember+ Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Lawo Ember+ configuration file named *AudioMixerLawoEmberPlus.xml*.

Please see these files for the available settings and their documentation.

Copyright © 2026 Vizrt

LAWO PowerCore Configuration File

This driver reads *AudioMixerEmberPlus.xml*.

Please see that file for the available settings and their documentation.

MidiMixer Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific MidiMixer configuration file named *AudioMixerMidiMixer.xml*.

Please see these files for the available settings and their documentation.

Stagetec Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Stagetec configuration file named *AudioMixerStagetecDiamondRas.xml*.

Please see these files for the available settings and their documentation.

Stagetec Ember+ Configuration File

This driver reads the Ember+ configuration file named *AudioMixerEmberPlus.xml*.

Please see that file for the available settings and their documentation.

Studer 3000 TCP/IP Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Studer 3000 TCP/IP configuration file named *StuderOnAirEmberConfiguration.xml*.

Please see these files for the available settings and their documentation.

Studer Vista Configuration File

This driver first reads the general configuration file named *AudioMixerConfiguration.xml*.

Then it reads the specific Studer Vista configuration file named *StuderVistaEmberConfiguration.xml*.

Please see these files for the available settings and their documentation.

Tascam Sonicview 16 Ember+

This driver reads the Ember+ configuration file named *AudioMixerEmberPlus.xml*.

Please see that file for the available settings and their documentation.

General Audio Mixer Configuration File

Audio mixers not specifically mentioned above, all use the general configuration file named *AudioMixerConfiguration.xml*.

Please see that file for the available settings and their documentation.

Copyright © 2026 Vizrt

Info: In the default version of this file, there are some settings specific to the Lawo RMNOPL audio mixer driver.

Engine Switcher Configuration File

For setup and configurations that use these settings, please refer to section Viz Engine as a Switcher.

- switcher_config.cfg
- ClipServerVizEngine.xml

switcher_config.cfg

Info: These configurations are performed on the Viz Engine (not Viz Mosart) machine.

File Location

The Switcher Scene can be configured from its configuration file, `switcher_config.cfg` located in C:\ProgramData\vizrt\VizEngine.

Configurable Parameters

#scene path for preview engine
<switcher_scene>Vizrt\Engine-Switcher\switcher</switcher_scene>

#scene path for openers if static_paths = 0 or 2
<opener_path>Vizrt\Engine-Switcher\openers\</opener_path>

##scene path for effects if static_paths = 0 or 2
<effect_path>Vizrt\Engine-Switcher\effects\</effect_path>

#type of paths: 0 = absolute path i.e. ALL/MyPath/openers, 1 = openers/effects underneath switcher scene, 2 = relative path underneath switcher scene i.e. myeffects
<static_paths>2</static_paths>

#amount of effect scenes for preloading
<effects>10</effects>

#content transition effect number to begin with
<content_effect_start>90</content_effect_start>

#external transition effect number to begin with, end at content_effect_start
<ext_effect_start>50</ext_effect_start>

#external opener/trans within GFX channels
<gfx_channel_support>1</gfx_channel_support>

#activating external transition effect engine - not recommended due to possible network delays
<ext_trans_active>0</ext_trans_active>

#transition engine ip
<transition_host>127.0.0.1</transition_host>

#transition engine port
<transition_port>6100</transition_port>

#activating external top layer engine
<ext_top_scene_active>0</ext_top_scene_active>

#external top layer engine ip
<ext_top_scene_host>127.0.0.1</ext_top_scene_host>

#external top layer engine port
<ext_top_scene_port>6100</ext_top_scene_port>

#activating external preview
<ext_prev_active>0</ext_prev_active>

#external preview hostname
<ext_prev_host>127.0.0.1</ext_prev_host>

#external preview port
<ext_prev_port>6100</ext_prev_port>

#show commands (log level)
<show_commands>2</show_commands>

#delaying the actual take in fields coming via external gfx scenes - this is used for having more time to load ip sources in preview
<external_take_delay>0</external_take_delay>

#i am peview
<i_am_preview>0</i_am_preview>

#activating dynamic ip workflow (Probel source request)
<dynamic_ip_workflow>0</dynamic_ip_workflow>

#shared memory map name used for status signaling
<dynamic_ip_status>pgmstatus_prv</dynamic_ip_status>

#shared memory map name used for Probel status callbacks - mandatory for using Probel communication!
<dynamic_ip_callback>probelstatus-machinename</dynamic_ip_callback>

#shared memory map name used for indicating a toggle (might be used in other gfx scenes)
<toggle_status>toggle_mapname</toggle_status>

#time waiting for the release of "old" sources after the toggle of new ones
<release_delay>10</release_delay>

#Source Prefix
<source_prefix>LIVE</source_prefix>

#size of the maximum number of active preview sources
<preview_queue_size>5</preview_queue_size>

#viz one in use (needed for setting clip prefix)
<viz_one>0</viz_one>

#clip prefix
<clip_prefix>V:\</clip_prefix>

#defines how the status overlay scene gets loaded - 1 = direct load to frontlayer - 2 = dynamic scene container in fl scene
<status_overlay_mode>1</status_overlay_mode>

#path to status overlay scene in GH
<status_overlay>Default\statusoverlay_new</status_overlay>

#start playing clip on preview machines
<play_clip_on_preview>0</play_clip_on_preview>

Copyright © 2026 Vizrt

#trigger tally lights
<tally_active>0</tally_active>

#tally server name
<tally_name>tally_pcr_main</tally_name>

#tally backup server name
<tally_name_backup>tally_pcr_backup</tally_name_backup>

#custom content transition effects
<content_transitions>80:wipe,81:roll,82:twist</content_transitions>

#vs sources input table (not used)
<vs_input_sources>11:960,12:961,13:962,14:963,15:964,16:965,17:966,18:967,21:960,22:961,23:962,24:963,25:964,26:965,27:966,28:967,31:960,32:961,33:962,34:963,35:964,36:965,37:966,38:967</vs_sources>

#vs sources output table (not used)
<vs_output_sources>11:42,12:43,21:44,22:45,31:46,32:47</vs_sources>

#tracking hub hosts (not used)
<tracking_hub_A>10.29.33.54:20000</tracking_hub_A>
<tracking_hub_B>10.29.33.54:20000</tracking_hub_B>
<tracking_hub_C>10.29.33.54:20000</tracking_hub_C>

#delay added for fullscreen toggle - this helps in adjusting proper timing with VS switches
<full_toggle_delay>4</full_toggle_delay>

#delay for VS switch when toggling from a fullscreen to a window with a cam switch happening
<full_win_delay>4</full_win_delay>

#time waiting for toggling superchannels when an external wipe animation is triggered
<wipe_delay>20</wipe_delay>

#time waiting for toggling superchannels containing clips which need to match a wall clip
<wall_clip_delay>10</wall_clip_delay>

#gpi sharedmemory prefix name
<gpi_shm_name>Sealevel::0::I</gpi_shm_name>

#tracking hub user
<th_user>user1</th_user>

#CUE Clip Workflow
<cue_clip_on_load>1</cue_clip_on_load>

#replacing clip path when file is not found
<substitute_clip>V:\CLIP_NOT_READY.mxf</substitute_clip>

#shared memory map name used for storing the upcoming label names
<label_names>MIXER_LABELS</label_names>

ClipServerVizEngine.xml

This configuration is employed on the Viz Mosart side, when controlling video switching directly from a Vizrt Viz Engine, as presented in section Viz Engine as a Switcher.
For available configuration options, refer to section Vision Mixer Configuration Files.

Graphics Configuration Files

This section contains the graphics type:

- Pixel Power Control Center (PPCC) Configuration File
- Pixel Power (Clarity) Configuration File
- Vizrt Graphics Configuration File
- Vizrt Media Sequencer VDom Logic Macros
- Chyron Configuration File
- Deko, Orad, Xpression Configuration File

Pixel Power Control Center (PPCC) Configuration File

The PPCC configuration file is called *PPCCConfiguration.xml*

```xml
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="PixelPowerConfiguration" connectionString="localhost:0">
  <Properties>
    <!-- Default channel nubmer to use -->
    <item name="SystemNumber" value="1"/>
    <item name="ClientApplication" value="Overlay Graphics"/>
    <item name="ClientName" value="Vizrt Mosart"/>
    <item name="DefaultOverlaysTriggerCount" value="2"/>
    <item name="EnableOutputBlackOnTakeOut" value="true"/>
    <item name="PreventPreloadOverlayIfItemsAreOnAir" value="true"/>
    <item name="RundownPrefix" value=""/>
    <item name="ConvertStoryIdToDecimal" value="false"/>
    <item name="ConnectedWhenIdle" value="false" />
    <item name="ConnectedWhenInStandBy" value="false" />
  </Properties>
</DeviceConfig>
```

Properties

- **DefaultOverlaysTriggerCount:** Number of triggers used for take-in and take-out from PPCC in overlays.
- **EnableOutputBlackOnTakeOut:** True if OutputBlack should be sent when continue points still exist on item.
- **PreventPreloadOverlayIfItemsAreOnAir:** Will prevent selecte page of next overlay if the engine currently has items on air (only overlays).
- **RundownPrefix:** The prefix to be used in the roId of the commands. (The same as set in the mosconfig.xml file in the Avid MOS Gateway.
- **ConvertStoryIdToDecimal:** Specifies whether the story id should be converted from Hex to Decimal value.
- **ConnectedWhenIdle:** This value determines if the connection to the graphic devices are maintained while Viz Mosart Server is in Idle mode.
  True = Maintain connection to graphic devices while in Idle mode. This setting will also initialize a connection to devices if Viz Mosart Server is started in Idle.
  False = Do not start/maintain connection while in Idle mode.

- **ConnectedWhenInStandBy:** This value determines if the connection to the graphic devices are maintained while Mosart is in Standby mode.
  True = Maintain connection to graphic devices while in Standby mode. This setting will also initialize a connection to devices if Mosart is started in*Standby*.
  False = Do not start/maintain connection while in Standby mode.

Note:

- You can specify Channel on the PPCC for each PPCC engine in AV Automation and Overlay Graphics.
- This channel will only be used for OutputBlack messages (clearing an engine).
- It's possible to define *multiple* channels using both comma and semicolon as separator.
- Be aware that the channel will be cleared on the PPCC regardless of the connection information. For example, clearing channel 1 clears *all* graphics on this channel.

Pixel Power (Clarity) Configuration File

The Pixel Power (Clarity) configuration file is named *PixelPowerConfiguration.xml*.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="PixelPowerConfiguration" connectionString="localhost:0">
    <Properties>
        <!-- Connection parameters, may be overridden -->
        <item name="Server" value="localhost" /> <!-- Not in use -->
        <item name="Port" value="10220" /> <!-- Not in use -->
        <item name="DefaultTakeOutPage" value=""/> <!-- Not in use -->
        <item name="OutputBlack" value="false"/> <!-- Not in use -->
        <item name="JobPath" value="c:\Pixel_Power\Jobs\"/> <!-- Not in use -->
        <item name="Extension" value=".pjz"/>
        <item name="LowPageRange" value="1-999"/> <!-- Not in use -->
        <item name="FullScreenPageRange" value="1000-1999"/>
        <item name="CgPageRange" value="2000-9999"/>
        <item name="ClarityProtocolVersion" value="1.0"/>
        <item name="UseAsUpdateJob" value="false"/>
        <item name="CreatePageMode" value="true"/>
        <item name="WaitHandle" value="2000"/>
        <item name="HeartbeatInterval" value="10"/>
        <item name="PathCriteria" value=":"/>
        <item name="DisableAVAutomationLoadJob" value="true"/>
        <item name="DefaultOverlayPage" value="999" />
        <item name="UseDefaultPageChannel" value="true" />
        <item name="WaitForStories" value="3000"/>
        <item name="AskPageInfo" value="false"/>
        <item name="AskPageImage" value="false"/>
        <item name="AskFieldInfo" value="false"/>
        <item name="ConnectedWhenIdle" value="false" />
        <item name="ConnectedWhenInStandBy" value="false" />
    </Properties>
</DeviceConfig>
```

Note: Items that are no longer supported for Viz Mosart's Pixel Power implementation are noted with `<!-- Not in use -->`.

Properties

- **Extension:** The file extension for the Pixel Power job to be loaded.
- **FullScreenPageRange:** Defines the range of Pixel Power page numbers used by Viz Mosart for full screen pages.
- **CgPageRange:** Defines the range of Pixel Power page numbers used by Viz Mosart for CG (lower third) pages.
- **UseAsUpdateJob:** Set if Pixel Power is to treat the loaded job as an update job.
- **CreatePageMode:** Informs Pixel Power to create the page from Viz Mosart or not.
- **WaitHandle:** Defines how long Viz Mosart should wait before removing a layer from the cued page after sending a trigger animation char command to Pixel Power.
- **HeartbeatInterval:** Defines how often Viz Mosart should check if the connection to Pixel Power is valid. The value is given in seconds.
- **PathCriteria:** Defines criteria that must be met (if any) for a job path to be valid.
- **DisableAVAutomationLoadJob:** If this value is set to true, only Overlay Graphics will load jobs on Pixel Power. Note that the current implementation will not handle other job/page logic from AV Automation (such as delete pages). This value is expected to be true.
- **DefaultOverlayPage:** Defines the overlay page to use as default on which to add layers to. This is the CG page that will be transferred on taking overlay graphics.
- **UseDefaultPageChannel:** Lets the pages created use the default page channel defined in Pixel Power. If set to false, the pages created will use the channel defined in engine settings.
- **WaitForStories:** Defines how many seconds Viz Mosart should wait for a story before deleting old/non-existing pages (handles shortcut keys from Viz Mosart GUI).
- **ConnectedWhenIdle:** This value determines if the connection to the graphic devices are maintained while Mosart is in Idle mode.
  True = Maintain connection to graphic devices while in Idle mode. This setting will also initialize a connection to devices if Mosart is started in Idle.
  False = Do not start/maintain connection while in Idle mode.
- **ConnectedWhenInStandBy:** This value determines if the connection to the graphic devices are maintained while Viz Mosart is in Standby mode.
  True = Maintain connection to graphic devices while in Standby mode. This setting will also initialize a connection to devices if Viz Mosart is started in *Standby*.
  False = Do not start/maintain connection while in Standby mode.
- **AskPageInfo:** Defines whether Viz Mosart requests Pixel Power for page info when creating a page.
- **AskPageImage:** Defines whether Viz Mosart requests Pixel Power for page image when creating a page.
- **AskFieldInfo:** Defines whether Viz Mosart requests Pixel Power for a pages field info when creating a page.

Note: It is not necessary to ask for the last three items when creating a page.

Vizrt Graphics Configuration File

The Vizrt Graphics Configuration file is named *VizrtGraphicsConfiguration.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="VizrtGraphicsConfiguration">
    <Properties>
        <item name="default_effect_dsk" value=""/>
        <item name="default_effect_wall" value=""/>
        <item name="default_effect_full1" value=""/>
        <item name="allowUpdateOfOnAirItems" value="true"/>
        <item name="RemoveUnusedChannelsOutputs" value="true"/>
        <item name="SetConceptOnOutputChannel" value="true"/>
        <item name="TakeInCommand" value="" />
        <item name="TakeOutCommand" value=""/>
    </Properties>
</DeviceConfig>
```

Properties

Note: For a complete list of available properties, refer to the comments in *VizrtGraphicsConfiguration.xml*. This file is usually located under folder C:\channeltemplates.

- **default_effect_dsk:** Default effect to be used on the dsk handler (must be lower case)
- **default_effect_wall:** Default effect to be used on the wall handler (must be lower case)
- **default_effect_full1:** Default effect to be used on the fullscreen graphics engine 1 (must be lower case)
- **allowUpdateOfOnAirItems:** Allows an item to be updated even if it is onair (default = true)
- **RemoveUnusedChannelsOutputs:** When enabled, MSE Mosart Profile will synchronize to the Overlay Graphics Configuration, i.e. all outputs that are not used in Viz Mosart will be deleted. (default = true)
- **SetConceptOnOutputChannel:** Current Concept Override functionality sets a context environment variable named "alternative_concept" when scheduling an operation to the MSE. This variable will override any Concepts defined for an Output in the VCP/Trio profile editor. In a setup where there are multiple outputs assigned to a Channel where each output has Concepts assigned except the first, all outputs will run with the main concept. The following setting will set the Concept of all outputs under Mosart control in the Profile to the currently selected Concept and this will be the default behavior. The old way of sending the value in the environment for ScheduleElement can still be used by setting the property SetConceptOnOutputChannel to false. (default = true)
  The following **Viz Trio properties** are used when connecting to Viz Trio. They were originally stored in *VizTrioConfig.xml*. Viz Trio connections must use port 6200.
- **TakeInCommand:** Used to override take in commands. Use this if you require something other than `page:take`
- **TakeOutCommand:** Used to override take out commands if you require something other than `page:takeout`

Vizrt Media Sequencer VDom Logic Macros

It is possible to send custom commands to the Media Sequencer. This can be done by VDom logic which can run on the Media Sequencer as macros from Viz Mosart.

An optional `VizrtUserMacros.xml` file should be created and placed in the config files folder with the following format:

Macro file format:

```
<macros>
  <macro name="myMacroName">
    <!-- Place a single VDom element here, for instance <env /> -->
  </macro>
  <macro name="StartScene">
    <env>
      <viz>RENDERER*STAGE START</viz>
    </env>
  </macro>
</macros>
```

The macros are executed with the following variables:

Variable	Value	Comment
profile	/config/profiles/MOSART	
viz	The viz handler for the engine targeted in the MACRO call	
channel	The profile channel name for the engine targeted in the MACRO call	Only send if channel names are applied to elements

Chyron Configuration File

The Chyron configuration file is named *ChyronConfig.xml*.

The configurable parameters are documented with self-explanatory values in the sample below.

Note: For Viz Mosart 5.13.0 you must specify the encoding in *ChyronConfig.xml*

- to either *utf-8* or *utf-16*
- apply the same encoding standard in the Chyron system.

```
<DeviceConfig name="DeviceConfig">
  <Properties>
    <!-- Macro for taking out graphics on Chyron channel x -->
    <item name="TakeOutMacro1" value="BREAKOUT_A" />
    <item name="TakeOutMacro2" value="BREAKOUT_B" />

    <!-- Configure settings for taking out with transition graphics on Chyron channel
x for Intelligent Interface Rules Engine (IIRE).
    Specify the transition to be used to effect out the graphic. In the examples
below, 2 take out commands have been defined for port 1 and 2
```

```
    with tranistion "animate". {GraphicID} is a placeholder and must be given in the
specified format. -->
    <item name="IIRETakeOutMacro1" value="L\{GraphicID}\animate" />
    <item name="IIRETakeOutMacro2" value="L\{GraphicID}\animate" />

    <!--
         Encoding: Default encoding utf-16
         Page containing valid .Net encodings:
         http://blogs.msdn.com/b/shawnste/archive/2009/08/18/alternate-encoding-
names-recognized-by-net-ie.aspx
    -->
    <item name="Encoding" value="utf-16"/>
    <!-- If true the connection to the graphics devices are maintained while Mosart
is in idle mode -->
    <item name="ConnectedWhenIdle" value="false" />
    <!-- If true the connection to the graphics devices are maintained while Mosart
is in standby or idle modes -->
    <item name="ConnectedWhenInStandBy" value="false" />

    <!--Whether to send the NEW Check Existence command. (Requires Lyric build 2021
 or higher. Use Help / About Lyric to find out.)-->
    <item name="ShouldCheckExistence" value="false"/>

    <!--Timeout for the NEW Check Existence command-->
    <item name="TimeoutCheckExistence" value="1000"/>

    <!--
      Enable the PreviewOnOutput to keep the old behavior of cueing and taking an
item for preview.
      Leave as false for previewing on the local preview output of the Chyron. Note
that when the below
      setting is enabled, preview of CGs will be taken directly to air and is a
misconfiguration of the system.
    -->
    <item name="PreviewOnOutput" value="false"/>
  </Properties>
</DeviceConfig>
```

Deko, Orad, Xpression Configuration File

Several graphics drivers use variants of the common config file `IntelligentInterfaceConfiguration.xml`. These include

- **Deko**
- **Orad**
- **Xpression**

Their respective parameters are documented with self-explanatory values in the code below.

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<DeviceConfig name="IntelligentInterfaceConfiguration">
  <Properties>
    <!-- Config file for all Intelligent Interface Graphics Devices -->
    <item name="IgnoreCreateCommandInStandby" value="false" />
    <item name="LoadInBackground" value="false"/>
    <!-- Encoding to be used for communication. Typical values: "utf-16", "utf-8" -->
    <item name="Encoding" value="utf-16"/>
    <!-- The name of the Xml attribute containing the graphics id used when
communicating with the graphics devices -->
    <item name="GraphicsIdAttribute" value="graphics_id" />
    <!-- If true the connection to the graphics devices are maintained while Mosart
is in idle mode -->
    <item name="ConnectedWhenIdle" value="false" />
    <!-- If true the connection to the graphics devices are maintained while Mosart
is in idle mode -->
    <item name="ConnectedWhenInStandBy" value="false" />

    <!-- DEKO -->
    <item name="UseEffectsAttachedToTemplates" value="true"/>
    <item name="TakeInWithCueAndTransferKeyCodes" value="false"/>
    <item name="IgnoreKeyCodesOnTakeIn" value="false"/>
    <item name="ClearScene" value=""/>
    <!--<item name="TakeInPrefixFileName" value="\xF7"/>
    <item name="TakeInPostfixFileName" value="\xF8\xF7"/>-->

    <!-- XPRESSION -->
    <item name="XpressionResumeBeforeTakeOffline" value="false" />
    <!--true/false-->
    <item name="XpressionTakeOfflineDelay" value="2000" />
    <!--milliseconds (req. XpressionResumeBeforeTakeOffline to be true)-->
    <item name="XpressionSuppressAlfanumericTemplates" value="false" />
    <!--true/false-->
    <item name="XpressionRetryIndexGenerator" value="10" />
    <!--true/false-->
    <item name="XpressionOverlayIndexLimit" value="100000" />
    <!--true/false-->

    <!-- GRAPHICS : ORAD -->
    <!-- Activates preload in preview for given destinations.
         Syntax: comma separeted list of destinations: Example "DSK,WALL" -->
    <item name="OradPreloadDestinations" value="" />
    <!-- Either [1,2]. The number of response message expected after a preload
command V\5\13. Default 1-->
    <item name="OradPreloadNumResponses" value="1" />
    <!-- Orad Heartbeat interval in seconds. Set to 0 to disable -->
    <item name="OradPollInterval" value="60" />
    <!--seconds-->
    <!-- Minimum time to wait after take command before sending new commands to Orad
-->
    <item name="OradTakeBlocks" value="10" />
    <!--milliseconds-->
    <!-- Minimum time to wait after cue command before sending new commands to Orad
-->
```

```
    <item name="OradMaxWaitTillCued" value="6000" />
    <!--milliseconds-->
    <!-- Obsolete: Use general Encoding property instead -->
    <!--<item name="OradEncoding" value="unicode" />-->
    <item name="OradSkipRecue" value="never" />
    <!-- Either [last,loaded,never]. Controls when to skip recue-->
    <!-- If set, responses from last preview operation will be stored. Corresponding
play will not take place if cue is not verified -->
    <item name="OradEnableCueVerification" value="false" />
    <!-- If set, a Select Output command is sent as part of preview. Complient with
Orad/CII protocol, section 4.13-->
    <item name="OradEnableSelectOutput" value="false" />
  </Properties>
</DeviceConfig>
```

Light Control Configuration Files

This section contains the lighting control type:

- grandMA3 OSC
- ETC Eos Family OSC

grandMA3 OSC Configuration File

The grandMA3 OSC Configuration file is named *GrandMA3LightControl.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Properties for the GrandMA3 OSC light control driver-->
<DeviceConfig name="GrandMA3LightControl">
  <Properties>
    <!--Mosart continuously attempts to re-establish any lost or faulty connection to
the device.
    ReconnectInterval specifies the time (in ms) between each reconnection attempt.--
>
    <item name="ReconnectInterval" value="5000"/>
  </Properties>
</DeviceConfig>
```

ETC Eos Family OSC Configuration File

The grandMA3 OSC Configuration file is named *ETCEosLightControl.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Properties for the ETC Eos OSC light control driver-->
<DeviceConfig name="ETCEosLightControl">
  <Properties>
    <!--Mosart continuously attempts to re-establish any lost or faulty connection to
the device.
    ReconnectInterval specifies the time (in ms) between each reconnection attempt.--
>
    <item name="ReconnectInterval" value="5000"/>
  </Properties>
</DeviceConfig>
```

Copyright © 2026 Vizrt

Robotic Camera Configuration Files

This section contains the Robotic Camera types:

- Camerobot
- Shotoku
- FxMotion
- Vinten 200 Radamec
- Cambotics Configuration File
- Cinneo Configuration File
- Technodolly Configuration File
- Panasonic (serial) Configuration File
- Panasonic TCP Configuration File
- Electric Friends Configuration File

All configuration files are by default stored at `C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles\`. Please refer to the important Note in Device Configuration Files for information on copying the files before changing.

Camerobot

Specific Camerobot parameters are found in `RoboticCameraConfiguration.xml`.

- **SendDuration** (boolean) - If set to true, Camerobot sends the move duration as a `<duration>` element in its `MATRIX_CUT_MPOS` and `MATRIX_MOVE_MPOS` commands. The duration comes from the shot's Fade Time (Program or Preview) set in the Template Editor: a fade time of 0 stays an instant cut with no duration, while a non-zero fade time sends a move with the duration in seconds. Requires Camerobot protocol version 1.12 or later. Default *false*.
- **HeartbeatInterval** (integer) - Time between heartbeats, in seconds. If the property is absent, Camerobot falls back to `5`. The sample configuration file sets `10`.

Info: Both properties are studio-wide. They apply to every Camerobot controller configured in AV Automation, and cannot be set per controller or per camera. The configuration file is read when the driver connects, so restart AV Automation for a change to take effect.

Note: The Fade Time is entered in frames and converted to seconds using the configured video frame rate, rounded to two decimals. At 25 fps, a Fade Time of 50 frames sends `<duration>2.00</duration>`.

Note: Both properties are configurable from Viz Mosart 5.16. In Viz Mosart 5.15 the Camerobot driver read them from the connection string, which never delivered them, so `SendDuration` could not be enabled and `HeartbeatInterval` always used its default.

The Fade Time can also be made editable from the newsroom system. See *Camerobot Properties* in Template Device Functions.

Shotoku

Specific Shotoku parameters are found in `RoboticCameraConfiguration.xml`.

You can control behavior of the connection to Shotoku when switching Viz Mosart from main to backup, or at an AVA restart. Ensure the following properties are managed in the configuration file:

- **ShotokuForwardStandbyCommands** (boolean) - If set to true, standby state will be forwared to Shotoku controller. Default *false*.
- **ShotokuDisableHeartbeat** (boolean) - If set to true, the heartbeat mechanism is disabled. Default *true*.
- **ShotokuDisconnectInStandby** (boolean) - If set to true, disconnect controller when in standby. Default *false*.

The above settings govern the two situations:

1. Whether the sending of a disable command (when AVA restarts or when the active Mosart server is switched) to the Shotoku controller will block all automation commands until an enable command is received.
2. Whether instead of timing-out, Mosart shall stay connect to the Shotoku controller when in standby/idle.

In addition, the `HeartbeatInterval` property also applies to Shotoku. Refer to the configuration for file for information.

FxMotion

Specific FxMotion parameters are found in `RoboticCameraConfiguration.xml`. Refer to the configuration for file for information.

- HeartbeatInterval
- FxMotionReceiveTimeout

Vinten 200 Radamec

Specific Vinten 200 Radamec parameters are found in `RoboticCameraConfiguration.xml`. Refer to the configuration for file for information.

- HeartbeatInterval
- Vinten200ShowFilterValue
- Vinten200DisconnectInStandby
- Vinten200MinSendInterval

Cambotics Configuration File

The integration between Viz Mosart and Cambotics Camera Robotics Control Systems supports moving to a stored position. Camera Robotics control is handled by AV Automation.

For details on the Cambotics (Ross CamBot) Protocol, refer to the product documentation written by Ross Video (www.rossvideo.com).

Copyright © 2026 Vizrt

The Cambotics configuration file is located in the program-folder under `\Mosart Medialab\Mosart server\ConfigurationFiles\CamboticsConfiguration.xml`.

The following configurations are available:

- **Heartbeat:** (keep connection alive) interval: Default `2000` ms.
- **Connection timeout:** Default `1500` ms.
- **ConnectionAttempts:** Default `5` tries.
- **ConnectionDelay:** Default `60000` ms.

Info: To integrate with Cambotics (Ross Cambot) Camera Robotics, you must also configure it in Camera Robotics. Select the router protocol CAMBOTICS, set the IP address of the camera controller, and the Port number (default is set to `2050`).

Info: For more information on setting up templates for Cambotics, see the section on *Robotic Camera Control* in the *Viz Mosart User Guide*.

Cinneo Configuration File

Cinneo configuration is contained within the general device configuration file *DeviceConfig.xml*.

```
<item name="cinneoSceneFileTimeout" value="30000" />
```

- **cinneoSceneFileTimeout:** The value is given in milliseconds.

Technodolly Configuration File

The Technodolly configuration file is named *TechnodollyConfig.xml*.

```
<DeviceConfig name="TechnodollyConfiguration" connectionString="localhost">
   <Properties>
      <!-- Connection parameters, may be overridden -->
      <item name="Port" value="15243" />
      <item name="HeartbeatInterval" value="20" />
      <item name="MaxQueueLength" value="2" />
      <item name="ProtectInPreview" value="cut,move" />
      <item name="ProtectInProgram" value="-" />
   </Properties>
</DeviceConfig>
```

- **Port:** Contains the default value, it is not used.
- **HeartbeatInterval:** Defines how often Viz Mosart should check if the connection to Technodolly is OK. The value is given in seconds.

Copyright © 2026 Vizrt

- **MaxQueueLength:** Restricts queuing of commands to Technodolly. A value of `1` means only one prepare or one run. A value of `2` means one prepare and/or one run. A value greater than `2` means no queuing restrictions.
- **ProtectInPreview:** Defines which commands should not be sent to Technodolly in preview when a mixer cross-point is protected. Valid values are "-" for none, "cut" for prepare, "move" for run, and "cut,move" for both prepare and run.
- **ProtectInProgram:** Defines which commands should not be sent to Technodolly in program when a mixer cross-point is protected. The values have the same meaning as for *ProtectInPreview*.

Panasonic (serial) Configuration File

The Panasonic configuration file (used for PANASONIC and PANASONICNEW) is named *PanasonicConfiguration.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="PanasonicConfiguration">
    <Properties>
        <!-- Connection parameters, may be overridden -->
        <item name="HeartbeatInterval" value="2000" />
        <item name="DisableHeartbeat" value="false" />
    </Properties>
</DeviceConfig>
```

- **HeartbeatInterval:** Time between heartbeats, in milliseconds. Integer >= 0. Default = `2`.
- **DisableHeartbeat:** If false, heartbeats are sent. This checks the connection and sends a keep alive message to the controller. If true, no heartbeats are sent. Default = False.

Panasonic TCP Configuration File

The Panasonic TCP configuration file is named *PanasonicTcpConfiguration.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="PanasonicConfiguration" connectionString="localhost">
  <Properties>
    <!-- Connection parameters, may be overridden -->

    <!--Whether to disable pings. If true, pings are still sent if no response is
received for a request.-->
    <item name="DisablePing" value="false" />

    <!--The path component of the Request-URI. Do not change.-->
    <item name="Path" value="cgi-bin/aw_cam" />

    <!--Time between pings in ms-->
    <item name="PingInterval" value="2000" />

    <!--In case of connection loss, a ping will be sent after this many ms-->
    <item name="ReconnectInterval" value="1000" />
```

```
    <!--The number of ms to wait for a response for a Ping command-->
    <item name="TimeoutPing" value="500"/>

    <!--The number of ms to wait for a response for a SelectPort command-->
    <item name="TimeoutSelectPort" value="10"/>

    <!--The number of ms to wait for a response for a PlayPreset command-->
    <item name="TimeoutPlayPreset" value="10"/>
  </Properties>
</DeviceConfig>
```

- **DisablePing**: Whether to disable pings. If true, pings are still sent if no response is received for a request.
- **Path**: The path component of the Request-URI. Do not change.
- **PingInterval**: Time between pings in ms.
- **ReconnectInterval**: In case of connection loss, a ping will be sent after this many ms.
- **TimeoutPing**: The number of ms to wait for a response for a Ping command.
- **TimeoutSelectPort**: The number of ms to wait for a response for a SelectPort command.
- **TimeoutPlayPreset**: The number of ms to wait for a response for a PlayPreset command.

Electric Friends Configuration File

The configuration file for Electric Friends should be saved as *ElectricFriendsConfiguration.xml* normally stored in *C:\ChannelTemplates*

> **Info:** For more information on setting up templates in AV Automation, see the section on *Robotic Camera Control* in the *Viz Mosart User Guide*.

Example configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceConfig name="ElectricFriendsConfiguration" connectionString="localhost">
    <Properties>
        <!-- Connection parameters, may be overridden -->
        <item name="HeartbeatInterval" value="2000" />
        <!-- Time between heartbeats in ms -->
        <item name="PollDataInterval" value="10000" />
        <!-- Time between renewing list of shows, cameras and shots, in ms -->
        <item name="UseJsonWorkaround" value="true" />
        <!-- Early development. Timed move command needs a workaround for correct
response to server. Please let the value remain true -->
        <item name="PollRunningStatus" value="200" />
        <!-- Time between polls in ms. This will check to see if camera movement is
finished before starting a new move -->
        <item name="PollTimeout" value="5000" />
        <!-- Time the poll is allowed to run before ignoring the play command, in ms
-->
        <item name="DefaultPort" value="9000" />
        <!-- If no port is set, use this value. The ElectricFriends default port is
9000 -->
    </Properties>
```

```
</DeviceConfig>
```

Subtitling Configuration Files

ScreenLL Configuration File

The ScreenLL configuration file is named *ClipServerScreenLL.xml*.

```
<?xml version="1.0" encoding="utf-8" ?>
 <MediaServerConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Properties>
     <!-- Language code to look for to decide clip availability -->
     <item name="AvailabilityLanguageCode" value="OPN" />
     <!-- First Incue Time code that indicates invalidated clip -->
     <item name="InvalidatedFirstIncueTime" value="--:--:--:--" />
   </Properties>
 </MediaServerConfig>
```

Properties

- **AvailabilityLanguageCode:** Specifies which language code must be present in order for a clip to be marked as available. The language code is a three letter text string, which relates to a list of language codes found in the file *PoliSTLServerLangs.lst*, which must be present on the ScreenLL subtitle file server.
- **InvalidatedFirstIncueTime:** Specifies the First Incue Time value that will indicate that a subtitle clip has been invalidated (is not available even if it matches the *AvailabilityLanguageCode)*.

Other configuration items are customer-specific or for Vizrt internal use only.

Copyright © 2026 Vizrt

Video Router Configuration Files

- Miranda NVision Configuration File
- EvertzQuartz

Miranda NVision Configuration File

The integration between Viz Mosart and Miranda NVision video routers supports setting cross points on the router.

- Viz Mosart will interface directly with the NVISION 5128 Router.
- Router Control is handled by AV Automaton.
- Testing can be done using the Viz Mosart TestRouterControl application.

The config file is called VideoRouterMirandaNV9000.xml, located at \\Mosart Medialab\Mosart Server\ConfigurationFiles

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="MirandaNV9000Config">
  <Properties>
    <!--The amount of time (in ms) to wait before reconnection after unsuccessful
connect or lost connection-->
    <item name="ReconnectInterval" value="10000"/>

    <!--The Take mode parameter to the Take commands. Refer to the NV9000 spec pp.
8-9 for details. The default value is Automatic take mode.-->
    <item name="TakeMode" value="0x00000001"/>

    <!--Not used-->
    <item name="TimeoutSetCrossPoint" value="1000"/>

    <!--The User ID. Available in the NV9000 DB. Alternatively the client IP address
as a network byte order (big-endian) 32 bit word. E.g., the (hex) IP address
FE.DC.BA.98 should be written
    0xFEDCBA98.-->
    <item name="UserID" value="0xFEDCBA98"/>

    <!--Whether (True/False) to use the Take Source To Destination command. If False,
use the Take Input To Output command instead. Default True.-->
    <item name="UseTakeSourceToDestination" value="True"/>
  </Properties>
</DeviceConfig>
```

Protocol Details

For details on the NV9000 protocol, refer to the GrassValley product documentation (https://community.grassvalley.com/support/s/).

File Location

The Miranda Nvision configuration file is located in the program-folder under `\Mosart Medialab\Mosart server\ConfigurationFiles\VideoRouterMirandaNV9000.xml`.

Properties

- **ReconnectInterval**: The amount of time (in milliseconds) to wait before reconnection after unsuccessful connect or lost connection. Default: `10000`.
- **TakeMode**: The Take mode parameter to the Take commands. Default: `0x00000001` (Automatic take mode).
- **UserID**: The User ID. This can be fetched from the NV9000 database, or can be created by converting the client IP address to a network byte order (big-endian) 32 bit word. Default: `0xFEDCBA98`.
  For example, the (hex) IP address FE.DC.BA.98 should be written 0xFEDCBA98.
- **UseTakeSourceToDestination**: This value determines which protocol commands to use for setting a crosspoint:
  - True = 0x0000 3000 *Take Source To Destination* (default).
  - False = 0x0000 3001 *Take Input To Output.*

Note:
- To integrate with Miranda NVision video routers, you must also configure it in Router.
  - Select the router protocol MIRANDA NV9000.
  - Set the IP address of the router, and the Port number (use `9193`).

EvertzQuartz

The config file is called VideoRouterEvertzQuartz.xml, located at \\Mosart Medialab\Mosart Server\ConfigurationFiles

Video Server Configuration Files

Information about *Video Server* Configuration Files can be found in section Video Server, MAM and Database Connection Strings.

Vision Mixer Configuration Files

The configuration files for the following vision mixers (switchers) are listed here for convenience. See the inline commenting for parameter explanation.

Note: Only changes to commented values, where indicated as variables, are supported!

- ClipServerVizEngine.xml
- GvgVideoSwitcherConfig.xml
- GrassValleyCPLVideoSwitcherConfig.xml
- RossVideoSwitcherConfig.xml
- SonyVideoSwitcherConfig.xml
- VizEngineSwitcherConfig.xml
- VizrtIpSystemsVideoSwitcherConfig.xml

ClipServerVizEngine.xml

Settings for the video server configuration file and in addition, for Viz Engine as a video server. Only commented values may be changed.

```
<DeviceConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <Properties>
    <item name="KeepCache" value="False" />
    <item name="RefreshOnUpdate" value="True" />
    <!-- The layer a switching scene is located in on. Only used for video servers,
for Engine Switcher (Viz Engine) this value is always "MIDDLE". Default: Back -->
    <item name="ControlSceneLayer" value="MIDDLE" />
    <!-- Shall always be true, do not change. Default: true -->
    <item name="UseScriptCommands" value="true" />
    <!-- The path prefix of the clips. Default: <clip_root> -->
    <item name="ClipNamePrefix" value="" />
    <!-- Clip extension (if not set, all types can be played, but needs to be
specified in the NRCS). Default: empty -->
    <item name="ClipNamePostfix" value="" />
    <item name="AssetToMediaMap" value="UseTitleAsSlug" />
    <!-- The version of the Switcher Scene. Used by Mosart to check the channel
status when connecting to a port. For versions after 1, it does not need Mosart to
check the status. It is dynamic. Default: 1 -->
    <item name="SwitcherSceneVersion" value="2" />
 </Properties>
```

Note: For Engine Switcher Mode configurations in the Viz Engine, see Engine Switcher Configuration File.

GvgVideoSwitcherConfig.xml

Used by drivers GV DD35, GV KAYENNE PRIMARY, GV KAYENNE SECONDARY, GV ZOKDIAK, GVG200, GVG4000, GVG4000 V2 and NOVA700

```
<DeviceConfig name="GvgVideoSwitcherConfig">
  <Properties>
    <item name="SendBreak" value="true" />
    <item name="EnableHeartbeat" value="true" />
  </Properties>
</DeviceConfig>
```

- **SendBreak**: All GVG 4000 protocol switchers requires a break before connecting (but there are switchers which do not need this).
- **EnableHeartbeat**: When no commands are issued, a heartbeat is sent to the vision mixer with an interval equal to the configured heartbeat interval.

GrassValleyCPLVideoSwitcherConfig.xml

Used by driver GV CPL.

```
<DeviceConfig name="GrassValleyVideSwithcerConfig">
  <Properties>
    <!-- Interval in seconds when to test for a valid connection when not connected
-->
    <item name="ConnectionCheckIntervalInSecondsNotConnected" value="5" />
    <!-- Interval in seconds when to test for a valid connection when connected -->
    <item name="ConnectionCheckIntervalInSecondsConnected" value="10" />
  </Properties>
</DeviceConfig>
```

RossVideoSwitcherConfig.xml

Used by drivers ACUITY and CARBONITE.

Note: This config replaces CarboniteVideoSwitcherConfig.xml.

```
<DeviceConfig name="CarboniteVideoSwitcherConfig">
  <Properties>
    <!-- The properties below is the virtual cross point mapping for internal Ross
Carbonite and Acuity sources. Alias can be anything. Value is the virtual assigned
crosspoint to be set in AV Config XML. Name is the RossTalk command for the item.
    -->
    <item alias="MediaStore1" value="1001" name="MS:1" />
    <item alias="MediaStore2" value="1002" name="MS:2" />
    <item alias="MediaStore3" value="1003" name="MS:3" />
    <item alias="MediaStore4" value="1004" name="MS:4" />
    <item alias="Black" value="1008" name="BK" />
    <item alias="MatteColor" value="1009" name="BG" />
    <item alias="ME1" value="1011" name="ME:1" />
    <item alias="ME2" value="1012" name="ME:2" />
```

```
    <!-- For Carbonite the MiniME related settings control the corresponding MiniMEs
regions on the switcher, while for Ross Acuity these are used to control the B side
of the corresponding mixer region. -->
    <item alias="MiniME1" value="1021" name="MME:1" />
    <item alias="MiniME2" value="1022" name="MME:2" />
    <item alias="MiniME3" value="1023" name="MME:3" />
    <item alias="MiniME4" value="1024" name="MME:4" />
    <item alias="MultiScreen1" value="1031" name="MSC:1" />
    <item alias="MultiScreen2" value="1032" name="MSC:2" />
    <!-- The default number of aux buses for Carbonite is 16 and for Acuity is 64 -->
    <item name="NumberOfAuxBuses" value="16"/>
  </Properties>
</DeviceConfig>
```

SonyVideoSwitcherConfig.xml

Used by drivers SONY, SONY BVS, SONY MVS, SONY SERIAL TALLY, SONY SERIAL TALLY IP, KAHUNA and KAHUNA IP.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="SonySwitcherConfig">
  <Properties>
    <!-- Truns the heartbeat mechanism on/off. -->
    <!-- <item name="EnableHeartbeat" value="false" /> -->

    <!-- Connect when Mosart Server is Idle. -->
    <!-- <item name="ConnectedWhenIdle" value="false" /> -->

    <!-- The number of milliseconds before a time-out occurs when a read operation
does not finish.
         If set less than zero the time-out will be infinite -->
    <!-- <item name="ReadTimeout" value="100" /> -->

    <!--
      The duration (in ms) to wait after the break character was written to the
output buffer.
      This is to make sure the break signal is sent completely before sending any
other data.
    -->
    <!-- <item name="AfterBreakDuration" value="2" /> -->

    <!-- ComPort2 is used for the SonySerialTallyDriver in split mode, using the
second serialport for the SCU connection -->
    <!-- <item name="ComPort2" value="COM23" /> -->

    <!--
      Select "Moxa NPort" in the dropdown for SONY SERIAL TALLY switcher,
      and use the hostname/IP of the Moxa NPort device along with the TCP port and
Cmd port numbers.
      Example: 192.186.1.123@4001:966 where 192.186.1.123 is the IP address, 4001 is
the TCP port, and 966 is the Cmd port.
    -->
```

```
    <!-- <item name="MoxaNPort" value="192.186.1.123@4001:966" /> -->

    <!--
      Similarly to the main serial port, setting the value of ComPort2 to "Moxa
NPort" will use a second serial port of the Moxa NPort device for the SCU connection.
      The related MoxaNPort2 setting must also be set for this to work.
    -->
    <!-- <item name="ComPort2" value="Moxa NPort" /> -->

    <!--
      When ComPort2 is set to "Moxa NPort", this setting must be set to the IP
address and port of the second port of the Moxa NPort device.
      Example: 192.186.1.123@4002:967 where 192.186.1.123 is the IP address, 4002 is
the TCP port, and 967 is the Cmd port.
    -->
    <!-- <item name="MoxaNPort2" value="192.186.1.123@4002:967" /> -->

    <!--
      The duration (in ms) to wait after the baud rate change.
      Used only when MoxaNPort or MoxaNPort2 is configured.
    -->
    <!-- <item name="MoxaNPortAfterChangeDuration" value="1" /> -->


    <!-- The properties below routes the different commands sets to either the
standard or the extra COM port
         None: always com port 1,
         Partial: com port 1 if the crosspoint range or mix effect range is supported
by com port 2, otherwise com port 2
         Full: always com port 2
    -->
    <!--
    <item name="AutoTransComPort2" value="None" />
    <item name="SetCrossPointComPort2" value="Full" />
    <item name="SetCrossPointComPort2" value="Full" />
    <item name="SetAuxCrossPointComPort2" value="Full" />
    <item name="SetKeyFillCrossPointComPort2" value="Full" />
    <item name="SetKeyerComPort2" value="Full" />
    <item name="SetDskComPort2" value="Partial" />
    <item name="SetTransTypeComPort2" value="Full" />
    <item name="RecallRegionComPort2" value="None" />
    -->
  </Properties>
</DeviceConfig>
```

VizEngineSwitcherConfig.xml

Used by driver VIZRT ENGINE.
This config is employed when controlling video switching directly from a Vizrt Viz Engine, as presented in section Viz Engine as a Switcher.

```
<DeviceConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Properties>
    <item name="ControlScenePath" value="Vizrt/VizMosart/ControlScenes/switcher" />
    <item name="NumberOfKeyBuses" value="8"/> <!-- Number of key buses in template
editor (default value is 4, only other valid option is 8) -->
  </Properties>
</DeviceConfig>
```

Note: Configurations are also required on the Viz Engine server. See Engine Switcher Configuration File.

VizrtIpSystemsVideoSwitcherConfig.xml

Used by driver TRICASTER.

```
<DeviceConfig name="VizrtIpSystemsVideoSwitcherConfig">
  <Properties>
    <!-- The path where EMEMs are stored -->
    <item name="PathToEmem" value="D:\Sessions\SessionName\emem\" />

    <!-- Set this to true if the crosspoint names should be sent in the command to
the driver instead of the physical inputs for the NDI Outputs. Default: true.-->
    <item name="UseOutputCrosspointNames" value="true" />

  </Properties>
</DeviceConfig>
```

5.6.3 Video Server, MAM and Database Connection Strings

A *connection string* contains parameters needed for connecting a studio device to Viz Mosart. The string always comprises two parts:

- Standard Connection String Properties
  For example *Name*, *Type*, *Server*, *Config*.
  Config is the name of an associated configuration file that contains additional, custom settings.
- Device-specific Connection String Parameters
  Driver-specific configurations. For example, what to log in error situations, manipulation of standard commands, or the timeout behavior.

Connection strings are used by Viz Mosart components *Media Administrator* and, in some cases, *AV Automation*.

In AV Automation:

- Connection strings get auto-generated when configured in AVAutomation > Devices > Properties > Video Servers.
- When entering additional device parameters, to match specific control room parameters.

This section describes the following connection string types:

- System-level Location of Device Configuration Files
- Standard Connection String Properties
  - Viz Mosart Video Server Connection String
  - General Configuration File Properties
  - General Search Configuration File Properties
- Device-specific Connection String Parameters
  - AirSpeed Classic Connection String
  - AirSpeed Multi Stream / AirSpeed 5000 Connection Strings
  - AirSpeed Multi Stream Search
  - Amadeus Connection String
  - AMP Connection String
  - EVS IPDirector Connection String
  - EVS LinX Connection String
  - EVS Xedio Connection String
  - Grass Valley AMPP Connection String
  - Grass Valley K2 Connection String
  - Harris Nexio Connection String
  - JupiterWebService Connection String
  - MVCP Connection String
  - Omneon Connection String
  - Omnibus OPUS Connection String
  - Orad OCIP Connection String
  - Quantel Connection String
  - Vizrt TriCaster Configuration
  - ScreenLL Connection String
  - SimpleTestPlayer Connection String
  - SQL Database Connection String

Copyright © 2026 Vizrt

- VDCP Connection String
- Viz Engine Connection String
- Viz One and Media Service Connection String

System-level Location of Device Configuration Files

In a connection string where no path is provided as a value to the parameter *Config*, Viz Mosart searches for the file in the following order of folders:

- `%ProgramData%\Mosart Medialab\ConfigurationFiles\`
- `C:\ChannelTemplates\`
- `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\`
- `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\ConfigurationFiles\`
- `{Current folder}\ConfigurationFiles\`

Standard Connection String Properties

Irrespective of device type, several values are standard parameters:

- Viz Mosart Video Server Connection String
- General Configuration File Properties
- General Search Configuration File Properties.

Viz Mosart Video Server Connection String

Parameters not recognized by the driver in question are simply ignored.

```
Name=myServerName; Type=myServerType; Server=myServerAddress; Config=myConfigFile.xml
```

- **Name:** A logical name consistently associated with the device, allowing the device to be uniquely recognized throughout the system setup.
  All Media Administrator connection strings contain a *name*, which forms a distinction for Media Administrator, when handling multiple clip metadata sources and video servers.
  If the same video server is also controlled by AV Automation, *Name* in AV Automation must be the same as *Name* used by Media Administrator.

> **Info**: *Naming* plays a critical role in the powerful feature *Dynamic Video Server Configuration*.
> See the topic Dynamic Video Server Configuration.

- **Type:** Specifies a driver to be used for connecting to the server. Potential values are:
  - **AirSpace:** AirSpace driver.
  - **AirSpeed:** AirSpeed driver.
  - **AirSpeed Search:** As AirSpeed driver, but contains search functionality only.
  - **AirSpeedMultiStream:** AirSpeed Multi Stream driver, also Airspeed 5000.
  - **AirSpeedMultiStreamSearch:** As AirSpeed Multi Stream driver (above), but only search functionality.
  - **Amadeus:** Amadeus MAM driver.

    - **AMP:** AMP video server driver.
    - **AMPP:** Grass Valley AMPP video server driver.
    - **EvsIPDirector:** EVS IPDirector
    - **EVS LinX:** EVS LinX driver.
    - **EVSXedio**: EVS Xedio driver, used by AV Automation.
    - **EVSXedioSearch**: EVS Xedio driver for search functionality, used by Media Administration.
    - **GrassValleyK2:** Grass Valley K2 driver.
    - **HarrisNexio**: Harris Nexio driver.
    - **Jupiter Web Service**: Jupiter web service driver.
    - **MVCP**: MVCP driver.
    - **OPUS:** Omnibus MAM driver using OPUS interchange.
    - **Omneon:** Omneon (Harmonic) driver.
    - **OmneonSearch:** As Omneon driver, but contains search functionality only.
      Used by Media Administrator to obtain clip status from an Omneon (Harmonic) video server.
    - **Orad OCIP**: Orad OCIP and Avid FastServe driver.
    - **Quantel:** Quantel driver.
    - **ScreenLL:** Screen LinguaLib search driver.
    - **SimpleTestPlayer**: SimpleTestPlayer driver
    - **SqlMediaSearch**: SQL database driver.
    - **TriCaster**: Vizrt TriCaster driver used by AV Automation and Media Administrator.
    - **VDCP**: VDCP driver over serial port.
    - **VDCPtcp**: VDCP driver over TCP/IP port.
    - **VizEngine**: Viz Engine driver.
    - **VizOne**: Vizrt's Viz One and Media Service.
- **Server:** Specifies the server name or IP address of the video server.
- **Config:** Specifies a configuration file containing custom configuration for a particular video server driver. Editing these files is typically done during installation.
- **Static**: (Effective for *Media Administrator* only) If *True*, Media Administrator always connects to the MAM system (or video server), also when **Enable dynamic configuration** (see the topic Dynamic Video Server Configuration) is *True*, and the MAM system (or video server) is *not* part of the salvo selected in AV Automation (see also section *Virtual Server Ports* in Automation Devices Video Servers).

> **Info**: Normally, a MAM system will not be part of any salvo. This setting is mostly intended for MAM systems which should always be connected, regardless of the selected salvo. Default: *False*.

General Configuration File Properties

- **VerifyClipDelay** (integer): Delay in frames to wait before verifying new clips from the server.
- **KeepCache** (boolean): When set to *true*, the internal Media Administrator clip cache is kept on reconnect (Default: *false*).
- **RefreshOnUpdate** (boolean): If set to *true*, any clip update message received from the video server results in a clip refresh operation (Default: *false*).
- **ReleasePortOnDispose** (boolean): If *true*, the assigned video port is released when server connection is closed (Default: *true*).

- **SupportSearchUsingObjSlug** (boolean): Activates searching using *objSlug*. ObjSlug is the clip slug as shown on **Search** tab in Viz Mosart GUI's **Media Pool** (Default: *true*).
- **SupportSearchUsingRefID** (boolean): Activates searching using *RefID*. RefID is a specific clip property defined as a reference (Default: *false*).
- **VerifyModifiedClips** (boolean)**:** If *true*, clip modified/update events for all clips trigger a complete verification (for example treated as the clip was created). If *false*, only clips currently in the cache are verified. (Default: *false*).
- **CueTimeout** (integer): Specifies the maximum time in milliseconds for AV Automation to wait for a cue operation to complete (Default: *1000 msec*).
- **PlayTimeout** (integer): Specifies the maximum time in milliseconds for AV Automation to wait for a play operation to complete (Default: *0*).
- **SetLoopDelay** (integer): Specifies the time in milliseconds when to set the clip to play in loop after a successful cue operation. (Default: *1000 msec*).
- **NextPingDelay** (integer): Specifies the interval (in milliseconds) at which AV Automation issues a 'Ping' operation (verifying that the connection is still OK) against the server. The nature of the ping depends on the driver chosen.
  To modify the ping interval for Media Administrator see Media Administrator Properties Editor (**Ping Delay**).
- **NextServerAttemptDelay** (integer): In case of failure, specifies the time in milliseconds before trying a reconnect to the video server. (Default: *5000 msec*).
- **IgnoreCase (boolean):** If *true*, clip name searching is case insensitive. Must be used in combination with <MatchExpression ignoreCase="true" /> from General Search Configuration File Properties.
- **AssetToMediaMap**: Choose from two ways of mapping Asset to MediaObject. Either *UseTitleAsSlug* or *UseMamIdAsSlug*.
- **MaxClipNameLength**: Naming standard for recordings.
- **ClipNamePattern:** A pattern used for generating the names of recorded clips. The pattern may contain placeholders. Any placeholders are to be placed within braces ({…}). The following placeholders are available:

Placeholder	Substituted by
{clipname}	The value given in the recording dialogues, or from custom template commands.
{gallery}	The name of the gallery.
{templateset}	The name of the currently selected template set.
{timestamp}	The current time. This value can be formatted using the patterns from MSDN.

Example
Assume that the value of **ClipNamePattern** is {gallery}_{templateset}_{clipname}_{timestamp:yymmdd}
Here,
- The gallery is *MyGallery*
- The current template set is *MyTemplateSet*

- The clipname (given in the recording dialogues, or from custom template commands) is *MyClipName*
- The current time is *21 September 2016 12:00:00*

Then the name of the recorded clip will be *MyGallery_MyTemplateSet_MyClipName_160921*.

General Search Configuration File Properties

- **HitSortOrder** (enum): Specifies how to sort a list of multiple hits for a single search. Used in combination with the **BestMatchMethod**.
  Options:
  - **Ascending:** The list of hits are sorted ascending alphabetically.
  - **Descending:** The list of hits are sorted descending alphabetically.
  - **Unspecified** (Default): No sorting is applied to the resulting hits.
    Example: <HitSortOrder>Descending</HitSortOrder>
- **BestMatchMethod** (enum): Method used to determine single search result when a search returns multiple hits. Options:
  - **First:** Hits are sorted alphabetically and the first hit is returned.
  - **Last:** Hits are sorted alphabetically and the last hit is returned.
  - **Filter** (Default): Returns the first hit evaluated *true* from a *MatchExpression* filter (see below).
- **MatchExpressions** (list): List of filters to be used for narrowing searches that may result in multiple hits. Each MatchExpression filter is constructed from three attributes:
  - **ignoreCase:** (boolean) *True* if the filter is case insensitive. Must be used in combination with property **IgnoreCase** described in General Configuration File Properties.
  - **expression:** (string, regex) Expression used to create a regular expression then applied to the given clip name. Use the pattern attribute to extract parts of the original clip name to search for. (Default: *Empty,* meaning exact search).
  - **pattern:** (string) Regular expression used to extract information from the clip name.
    An empty **MatchExpressions** list results in an exact case sensitive search.
    (Default: *The original clip name*).

The following example creates regular expression filter searching:

MatchExpressions

```
<MatchExpressions>
    <!-- Returns true for all hits with clip names starting with the given clip name
-->
    <MatchExpression expression="^$1.*$"/>

    <!-- Returns true for all hits with identical clip name but case insensitive -->
    <MatchExpression ignoreCase=true/>

    <!-- Returns true for all hits with clip names containing leading or trailing
spaces -->
    <MatchExpression pattern = "^\s*(.*)\s*$" expression = "$1"/>
</MatchExpressions>
```

Device-specific Connection String Parameters

AirSpeed Classic Connection String

Applies to Models

- AirSpeed
- AirSpace

Connection String Properties

- **Type:** AirSpeed and AirSpace.
- **Name: Name** in Standard Connection String Properties above. The ID of this connection (Default: *AirSpeed*).
- **Server:** The hostname or IP-address of the server.
- **Config:** The name of the configuration file.

Examples

AV Automation:

```
Name=HAR; Type=AirSpeed; Server=192.168.1.1; Config=ClipServerAirspeed.xml
```

Configuration File Properties

Configuration file properties can be overridden by connection string properties.

- **KeepCache** (boolean): How often (in seconds) to look for clip changes. value=*1* is the default value if not configured here (Default: *false*).
- **RefreshOnUpdate** (boolean): When set to *true*, the cache is updated on each update clip message from the server (Default: *true*).
- **VerifyClipDelay** (integer): Delay in frames to wait before verifying new clips from server.
- **ReceiveTimeout** (integer): Time in seconds to wait for response from AirSpeed (Default: *10*).
- **ChangeCheckInterval** (integer): Time in seconds how often to poll AirSpeed for clip changes (Default: *2*).
- **HeartbeatInterval** (integer): Time in seconds to send out a heartbeat in case of no communication detected from AirSpeed (Default: *30*).
- **GetMinimumList** (boolean): Set to *true* to retrieve only clip names when initializing database (Default: *false*).
- **DelayGetClipData** (boolean): Delay getting detailed clip information. Information is retrieved for clips in rundown only (Default: *true*).
- **DelayEventsOnDatabaseInitialization** (boolean): If *true*, all clip handling during database load is postponed until last clip is received (Default: fa*lse*).
- **SignalDatabaseReady** (boolean): Refreshes all clips after database has been built instead of refreshing for every item in the database (Default: *true*).
- **VerifyModifiedClips** (boolean): If *true*, all modified clip events for clip not in cache are verified, as if they were created (Default: *true*).

Copyright © 2026 Vizrt

- **ReceiveBufferSize** (integer): Set buffer size used for receiving date from AMS. Default `16384` = two times default buffer size of Net.Socket (Default: *16384*).
- **Encoding** (string): Set to encoding used for send/receive from AMS (Default: *UTF-8*).
- **SupportSearchUsingObjSlug:** Ignores search for objSlug if set to *false* (Default: *true*).
- **SupportSearchUsingRefID:** Ignores search for refID if set to *false* (Default: *false*).

Use for debugging purposes only:

- **VerboseLogging** (boolean): Set to *true* to enable logging of received messages from AMS (Default: *false*).
- **DatabaseCache** (boolean): If set to a valid filename, initial load database stores the content in this file (Default: *false*).
- **DatabaseCacheUpdate** (boolean): Determines when the debug database cache is updated (either [*Never,Always,Initial*]).
- **LoadDatabase:** Specifies whether the AirSpeed clip database should be initially loaded and maintained by the Media Administrator.
  Setting this attribute to *true* is required if any search should be possible using an AirSpeed video server.

AirSpeed Multi Stream / AirSpeed 5000 Connection Strings

The AirSpeed Multi Stream drivers supports up to six player ports per connection.

Applies to Models

- AirSpeed Multi Stream 1.8
- AirSpeed 5000

Connection String Properties

- **Type:**
  - AV Automation: AirSpeedMultiStream
  - Media Administrator: AirSpeedMultiStreamSearch
- **Name:** See **Name** in Standard Connection String Properties above.
- **Server:** Defines the hostname or IP-address of the server.
- **Config:** See Standard Connection String Properties above. The installed xml config file is ClipServerMultiStream.xml. Unless changed, the value of **Config** should be *ClipServerMultiStream.xml*.
  See Examples below.

Optional

- **BasePort*:** Specifies the number of the lowest-numbered port on the server. (Default: *59451*). For AV Automation, six consecutive player ports, starting at this one, are available for playout. (They should be referred to by numbers 1 to 6 in AV Automation.)
  For Media Administrator, refer to Player and the note below for how the actual port number is calculated.
- **AV Automation only**
  - **LoopPostRoll:** (only used when *NewLooping* is *true*) No. of frames before clip-end to restart loop. Should probably be set to a positive value to obtain smooth looping. (Default: *4*).
  - **NewLooping:** Whether to use an alternative looping mechanism. Must be *true* if MPEG-2 HD clips are involved. (Default: *false)*.

- **OpenSomePlayerAfterConnect**: By default, server connection is considered successful ('green') without connecting to any players. If this setting is enabled, also connection to some configured player will be necessary. (Default: *false)*.
  Set to *true* if connection problems are experienced, e.g.
  i. Green server line
  ii. Unsuccessful port connect (red port line)
  iii. Reconnect
  iv. ... and so on ...
- **OpenPlayerAfterConnectTimeout**: (only used when *OpenSomePlayerAfterConnect* is *true*) Timeout (in milliseconds) for the above *OpenSomePlayerAfterConnect* functionality (Default: *2000*).
- **RecordEjectFirst**: If *true*, initiating a recording on a channel/port will first eject any existing clips (Default: *false*).
- **RecordOverwrite**: If *true*, any recording initiated on an existing clip will delete the existing clip before creating the new (Default: *true*).
- **RecordUseNameAsClipId**: If *true*, the clip name is also used as clip id when preparing a recording. Otherwise the Airspeed will automatically assign clip ids (Default: t*rue*).

Info: The asterisk (*) above denotes that the TCP port you connect to becomes BasePort + Player - 1.

Examples

AV Automation:

```
Name=AMS; Type=AirSpeedMultiStream; Server=192.168.1.1;
Config=ClipServerMultiStream.xml
```

Configuration File Properties

With a few exceptions (*Type*, *Name*, *Config*), the connection string properties described above can also be specified in the configuration file (keeping in mind that a connection string setting overrides its corresponding file setting).

In addition the following properties may be specified in the configuration file:

- **Media Administrator only**
  - **DelayEventsOnDatabaseInitialization** (boolean): If *true*, all clip handling during database load is postponed until last clip is received (Default: *false*).
  - **DelayGetClipData** (boolean): Delay getting detailed clip information. Information is retrieved only for clips in the rundown (Default: *true*).
  - **Encoding** (string): The encoding used for send/receive from AMS (Default: *UTF-8*).
  - **GetMinimumList** (boolean): When *true*, retrieve clip names only when initializing the database (Default: *false*).
  - **IgnoreCase:** Whether clip search is case-insensitive (Default: *false*).
  - **NumClips** (integer): Number of clips to receive for each server query (Default: *50*).
  - **ReceiveBufferSize** (integer): Set buffer size used for receiving date from AMS.
    16384 = two times default buffer size of Net.Socket (Default: *16384*).
  - **SignalDatabaseReady** (boolean): Refreshes all clips after database has been built instead of refreshing for every item in the database (Default: *true*).

- Use for troubleshooting only:
  - **DatabaseCache** (boolean): If set to a valid filename, initial load database stores the content in this file (Default: *false*).
  - **DatabaseCacheUpdate :** Determines when the debug database cache is updated (either *Never,Always* or *Initial*).
  - **VerboseLogging** (boolean): Set to *true* to enable logging of received messages from AMS (Default: *false*).
    Effective only if Verbose logging is turned on in Media Administrator.
- **AV Automation only**
  - **ReadTimeout:** Time in seconds to wait for response from AirSpeed (Default: *5*).

AirSpeed Multi Stream Search

This driver is similar to the AirSpeed Multi Stream driver above, but contains search functionality only.
It is used by Media Administrator for obtaining clip status from an AirSpeed Multi Stream or 5000 video server.

- **BasePort*:** The lowest port number to connect to (Default: *59451*).
- **Player*:** The player number (1-6). Used for setting the TCP port (Default: *1*).
- **LoadDatabase** (boolean): Set to true to build internal database of clips present on server and enable clip monitoring (Default: *false*).
- **ChangeCheckInterval:** Time in seconds how often to poll AirSpeed for clip changes (Default: *1*).
- **ReceiveTimeout:** Time in seconds to wait for response from AirSpeed (Default: *10*).
- **HeartbeatInterval:** Time in seconds to send out a heartbeat in case of no communication detected from AirSpeed (Default: *30*).

Info: The asterisk (*****) above denotes the TCP port to connect to becomes BasePort + Player - 1.

Example

```
Name=AMS; Type=AirSpeedMultiStreamSearch; Server=192.168.1.1; BasePort=59451;
Player=2; ChangeCheckInterval=5; ReceiveTimeout=5; HeartbeatInterval=20;
Config=ClipServerMultiStream.xml
```

Amadeus Connection String

```
Name=HTC-DB; Type=Amadeus; Port=[port]; Config=ClipServerAmadeus.xml
```

Connection String Properties

- **Name:** Amadeus ID for database. See **Name** in Standard Connection String Properties above.
- **Type:** Defines the Amadeus driver for clip handling.
- **Port:** Set to port to listen for an Amadeus connection.

Optional

- **Port:** Defines the connection port (Default: *2202*).
- **ResponseReadTimeout:** Defines the wait time until timeout (default: *5000*).
- **PollIntervalClips:** Defines the time between polling (Default: *15*). Set to *0* to disable.
- **RundownPrefix:** Defines the prefix to add for the rundown name (Default: "").
- **GetAllCGs:** If *true*, METAREQ_ALLCGS message type is used when getting clip info (Default: ""). For more details, see the Amadeus documentation.
- **LowerCaseRundowns:** If *true*, then the rundown name is always generated with lower case (Default: *false*).
- **IgnoreSTEG:** If *true*, all messages from Amadeus starting with STEG are ignored (Default: *true*). For more details, see the Amadeus documentation.
- **StegCommandIfSubtitled:** Sends the specified STEG command if clip is subtitled (Default: *KLOCKA INVOKEOFF*). For more details, see the Amadeus documentation.
- **SubtitleStegOutOn:** Sets the graphics_out_on xml value in the STEG command set to Amadeus (Default: *BACKGROUNDEND*).
- **PurgeCacheInterval:** Use hh:mm:ss format (Default: *00:00:00*).
- **PurgeCacheAge:** Use hh:mm:ss format (Default: *00:05:00*).

AMP Connection String

This connections string is used for video servers supporting the AMP protocol.

Connection String Properties

- **Type:** AMP.
- **Port:** 3811, unless another value has been specified by the manufacturer.
- **Config:** ClipServerAMP.xml (This file is included in the installation.)
- **LoadDatabase:** Use LoadDatabase=true for Media Administrator; *omit* for AV Automation.
- **Name, Server:** See Viz Mosart Video Server Connection String above.

Examples

AV Automation:

```
Name=AMPserver;Type=AMP;Server=...;Port=3811;Config=ClipServerAMP.xml
```

Media Administrator:

```
Name=AMPserver;Type=AMP;Server=...;Port=3811;Config=ClipServerAMP.xml;LoadDatabase=true
```

Driver-specific configuration file properties

Please see the installed ClipServerAMP.xml. The available properties are documented there.

EVS IPDirector Connection String

This connection string is used in Media Administrator to connect to the EVS IPDirector. It may *not* be used in AV Automation. In AV Automation, other kinds of connection strings should be used to connect to any video servers. A possible scenario will be to use EVS LinX video servers, see below.

> **Info**: This scenario (EVS IPDirector in Media Administrator and EVS LinX in AV Automation) is the only combination involving EVS IPDirector that is supported.

```
Name=[name];Type=EvsIPDirector;IpAddress=[IP
address];Login=[login];Password=[mosart];Port=31016;Config=ClipServerEvsIPDirector.xm
l
```

Connection String Parameters

- **Type**: EvsIPDirector.
- **Name**: See **Name** in Standard Connection String Properties above. The Name must be different from the Name of any other MAM system and video server, in both Media Administrator and AV Automation.
- **IpAddress:** The IP address of the IPDirector API.
- **Login:** The login to use.
- **Password:** The password.
- **Port:** The port of the IPDirector API (Default: *31016*).
- **Config**: Configuration file containing further configuration settings.

Config File Parameters

The Viz Mosart Server installation will contain a default config file *ClipServerEvsIPDirector.xml*. See this file for available parameters.
Any parameters set in this file will be overridden by the same parameters set in the connection string.

EVS LinX Connection String

This connection string is used for EVS video servers controlled by the LinX protocol (as opposed to EVS Xedio systems, see EVS Xedio Connection String).

```
Type=EVS LinX; Name=[name]; Server=[IPaddress]; LoadDatabase=true;
Config=ClipServerEvsLinX.xml
```

Example

```
Type=EVS LinX; Name=EvsLinXName; Server=192.168.62.6; LoadDatabase=true;
Config=ClipServerEvsLinX.xml
```

Copyright © 2026 Vizrt

Connection String Parameters

- **Type**: EVS LinX or EVS (*EVS* is retained purely for backward compatibility).
- **Name**: See **Name** in Standard Connection String Properties above.
- **Server**: Defines the IP address of the server.
- **LoadDatabase**: If *true*, loads all clip metadata into local cache (Default: f*alse*). Use *true* for Media Administrator.
- **Config**: Configuration file containing further configuration settings.

Optional

- **LogOutput**: The LinX log file created for the LogFlag option 0x00000003. The files for the other options are in the same folder, and with names derived from LogOutput (see **LogFlag** below for details.) The folder should exist in advance, and the Viz Mosart user should have permission to create and write files.
  AV Automation and Media Administrator should log to different files, so LogOutput should be specified (with different folders) in the respective connection strings rather than in this file.
- **VeryVerbose**: If *true* (and the Viz Mosart application issues verbose logging), then some extra log messages are issued (Default: *false*).

Config File Parameters

- **SupportSearchUsingObjSlug**: Search by 'slug' (clip name) is supported (Default: *true*).
- **SupportSearchUsingRefID**: Search by a configurable clip property (given by the SearchPropertyName parameter; see below) is supported (Default: *false*).
- **HeartbeatTimeout**: The time (in milliseconds) after which the server connection may close if there is no activity (Default: *12000*).
- **KeepAliveIntervalDivisor**: In case of connection problems, multiply both HeartbeatTimeout and KeepAliveIntervalDivisor by the same factor (for example, two) (Default: *8*).
- **LocalMtpcIpAddress**: Necessary only if the Viz Mosart server has several network adapters, e.g. a virtual adapter (or several virtual adapters) in addition to the physical adapter, e.g. when VMWare (or similar) is installed on a demo PC, or when connecting through VPN. In this case, use the IP address (in the usual format of four dot(.)-separated groups of one to three decimal digits) of the network adapter connected to the EVS servers.
- **LogFlag**: Bitmask specifying LinX log level. Available options are listed in the table below. (In addition, *linxFunctions.log* is produced.) The options may be combined. For example, 0xFFFFFFFF produces *all* log files.

Bitmask	Description	Log file, given LogOutput=...\linx.log
0x00000003	Function entry and exit points, with parameters and return code	linx.log
0x00000010	Connections	linxConnect.log

Bitmask	Description	Log file, given LogOutput=...\linx.log
0x00000200	Keeping alive	linxKeepAlive.log
0x00000400	Databases	linx_Archive.log
0x00008000	Debug	linxDebug.log
0x01000000	Tools	linxTool.log

- **MaxTries**: [Do not use! The maximum number of times to try to synchronize the database (Default: *10*)]. Obsolete. Instead, use **SharedNbMaxConnectRetry** and **SharedConnectTimeout**, as described below.
- **NbMaxLinxConnection**: The maximum number of connections. The default is *18*, to accommodate two EVS servers. Add *9* for each additional server.
- **OsdDefine00 - OsdDefine11**: Defines the 12 OSD lines.
- **OsdShowHideError**: Defines whether or not to show OSD.
- **OsdShowHideErrorSignals**: Defines whether or not to show OSD. (For further details of these and other OSD settings, please see the default configuration file).
- **SearchPropertyName**: Clip property that may be used for clip search. The default is *Varid* (a LinX specific clip ID that may be used to identify clips in the NRCS). If some other property is needed, please ask Viz Mosart personnel for a list of available values.
- **SharedNbMaxConnectRetry**: Maximum number of retries in case of connect error on a shared connection. Increase if (for example, when synchronizing the database) the error code *Too many retries* is encountered. (Default: *16*).
- **SharedConnectTimeout**: Maximum time in milliseconds to execute connection request on a shared connection. Consider increasing if the error code 'Too many retries' is encountered, e.g. when synchronizing the database. (Default: *500*).
- **SearchByVarID:** Whether search (from MediaAdmin or GUI) will use *Varid* (instead of the default, clip name) (Default: false)

Using LinX video servers together with IPDirector

As mentioned in EVS IPDirector above, EVS LinX video servers may be used together with IPDirector. In this case, Media Administrator connects to IPDirector, whereas AV Automation connects to *both* the LinX video server(s) and IPDirector. This will enable the 'remote playout' feature, making it possible to load and play on a server S1 a clip that does not reside on S1, but on another server, S2. (Only servers that will be used for playout (the *S1s*) need configuring in AV Automation. The *S2s* do not have to be configured.

To enable this feature,

- EVS LinX shall *not* be configured in Media Administrator. (So the configuration described here applies to AV Automation.)
- EVS IPDirector shall be configured in Media Administrator as described in EVS IPDirector, above.

- These parameters must be set (in the connection string or in the config file) for EVS LinX:
  - **UseIPDirectorAPI**: Set to true (Default: *false*).
  - **IpdIpAddress**: The IP address of the IPDirector API.
  - **IpdLogin:** The login to use.
  - **IpdPassword:** The password.
  - **IpdPort:** The port of the IPDirector API (Default: *31016*).

(The four Ipd… parameters should match the corresponding (without the 'Ipd' prefix) parameters in Media Administrator).

Additionally, this parameter may be set:

- **UseNtscNodrop**: Determines the kind of time codes to be used when loading clips if
  - the server video standard is not PAL
    and
  - the clip to be loaded has no video standard set.

  If *true*, NTSC No Drop timecodes will be used in these circumstances.
  If *false* (default), NTSC Drop timecodes will be used.

Applies to Models

- XS family (6U, 4U)
- XT family (XT2+, XT3, not nano)

Device Prerequisites

- Multicam 10.03, or later
- LinX license code
- Channels to be controlled by Viz Mosart must have Main CTRL LinX.
- The ports requirements depend on the Multicam version:
  - Multicam 20.7 and lower: The network, including any firewalls, connecting the Viz Mosart server(s) and the EVS server(s) must support and allow TCP (ports `50000` - `50002`) and UDP (unicast and multicast, ports 50001-50002 and `50100` - `50108`).
  - Multicam 21.0 and higher:
    - Server Listen Ports: TCP: 5970, UDP: 5971, 6150-6157.
    - Server Send Port: UDP: 5972.
- For the 'remote playout' feature to work, the video servers must be on the same XNet.

EVS Xedio Connection String

These connection strings are used for EVS Xedio video servers (as opposed to EVS video servers controlled by the LinX protocol, see EVS LinX Connection String, above).

From a Viz Mosart perspective, a Xedio system has three parts:

- One or more playout servers
- A control center
- A DB connected to through ODBC.

AV Automation controls the playout servers, whereas Media Administrator connects to the control center and the database. Consequently, the connection strings for AV Automation and Media Administrator are somewhat different.

AV Automation Connection String

```
Type=EvsXedio; Name=[name]; Config=ClipServerEvsXedio.xml
```

Example:

```
Type=EvsXedio; Name=EvsXedioName; Config=ClipServerEvsXedio.xml
```

AV Automation Connection String Parameters

- **Type:** EvsXedio.
- **Name:** See **Name** in Standard Connection String Properties above.
- **Config:** Defines the configuration file containing further details.

> **Info:** The parameters mentioned so far are insufficient to connect to a playout server. The IP address and port number are also required.
> These are specified (separated with **:**) in the Video port field(s) in AV Automation Device Properties.
> For example, *192.168.76.11:4021*.

Media Administrator Connection String

```
Type=EvsXedioSearch; Name=[name]; Server=[IPaddress]; Config=ClipServerEvsXedio.xml
```

Example:

```
Type=EvsXedioSearch; Name=EvsXedioSearchName; Server=192.168.76.10;
Config=ClipServerEvsXedio.xml
```

Media Administrator Connection String Parameters

- **Type**: EvsXedioSearch.
- **Name**: See **Name** in Standard Connection String Properties above.
- **Server**: Defines the IP address for the Control center.
- **Config**: Defines the configuration file containing further details.

> **Note**: Parameters mentioned so far are insufficient to connect to the database. Additional parameters *DataSourceName*, *User*, and *Password* are required.
> The defaults are probably sufficient. If not, these are more conveniently set as configuration file parameters, described below.

Configuration File Parameters

- **PrefixTypeEdit**: MOS objIDs with this prefix are treated as 'edits' (Default: *EVSE*).
- **PrefixTypeMedia**: MOS objIDs with this prefix are treated as 'media' (Default: *EVSM*).

Relevant for AV Automation Only

- **TimeOut**: Defines the timeout in milliseconds (Default: *1*).

Relevant for Media Administrator Only

- **AcceptHighResolutionOnly**: Whether only high resolution clips are accepted (Default: *false*).
- **DataSourceName**: ODBC System DSN entry for the CleanEdit DB (Default: *CleanEditDB*).
- **MonitorClipInterval**: The interval (in seconds) at which found clips are monitored (Default: *60*). Used only if MonitorFoundClips is *true*, see below.
- **MonitorFoundClips**: Whether found clips are monitored (Default: f*alse*).
- **Password**: Password for the CleanEdit DB.
- **PrefixCheckItemReturnAllItemHiResPresent**: The prefix added to the Description of a hi-res clip (Default: *HIRES-*).
- **PrefixCheckItemReturnError**: The prefix added to the Description of an error clip (Default: *ERROR-*).
- **PrefixCheckItemReturnNotPlayable**: The prefix added to the Description of non-playable clip (Default: N*OTPLAYABLE-*).
- **PrefixCheckItemReturnOnlyItemLowResPresent**: The prefix added to the Description of a lo-res clip (Default: *LORES-*).
- **SkipGetDuration**: Whether to skip getting clip duration. Note that this operation may be slow and MAY be rendered unnecessary by future Media Administrator dev (Default: *false*).
- **User**: User for the CleanEdit DB.

Debug Parameters, Relevant for Media Administrator Only

- **LogFile**: CleanEditOcx log file (Default=*C:\EvsLogs\CleanEditOcx\CleanEditOcx.log*).

Grass Valley AMPP Connection String

These connections strings are used for the Framelight X asset management system (GV AMPP, or Grass Valley Agile Media Processing Platform), which is a cloud-based service.

Connection String Properties

- **Type:** AMPP
- **PlatformURL:** The AMPP Platform URL. This URL should look like *https://<region>.cloud.grassvalley.com*
- **ApiKey:** An API key allowing you to connect to the AMPP Platform for sending/receiving AMPP Control messages
- **LocalPushSiteId:** The site id (a string of characters that uniquely identifies an area as a site), required for local push notifications.
- **LocalIpAddress:** The IP address, used to find local push notifications services.
- **LocalSubnetMask:** The subnet mask, used to find local push notifications services.
- **ResponseTimeout:** Timeout in milliseconds to wait for the responses from the server. Default 2000

- **QueryLimit:** For pagination when *LoadDatabase=true*. Limit the number of entries in the query results. Default value: 100
- **LoadDatabase:** Use LoadDatabase=true for Media Administrator; *omit* for AV Automation.
- **Site:** The name of the FramelightX (AMPP AM) site in which you wish to search for media; for Media Administrator only.
- **Stores:** The name of one ore more stores within the site configured above; for Media Administrator only.
- **Types:** Asset types to consider. If not set or empty, all assets will be fetched; for Media Administrator only.
- **Name:** See Viz Mosart Video Server Connection String above.

In most cases, the PlatformURL and ApiKey settings will be the same for AV Automation and Media Administrator.

Examples

AV Automation:

```
Name="GV AMPP";Type=AMPP;PlatformURL=https://
region.cloud.grassvalley.com;ApiKey="<API-key string>";
```

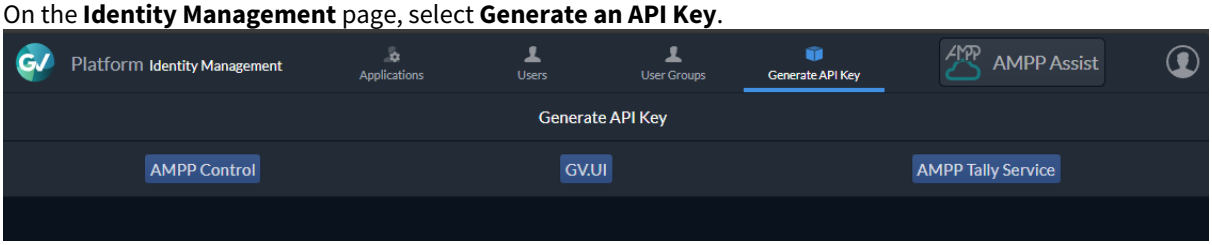
Media Administrator:

```
Name="GV AMPP";Type=AMPP;PlatformURL=https://
region.cloud.grassvalley.com;ApiKey="<API-key string>";LoadDatabase=true;
```

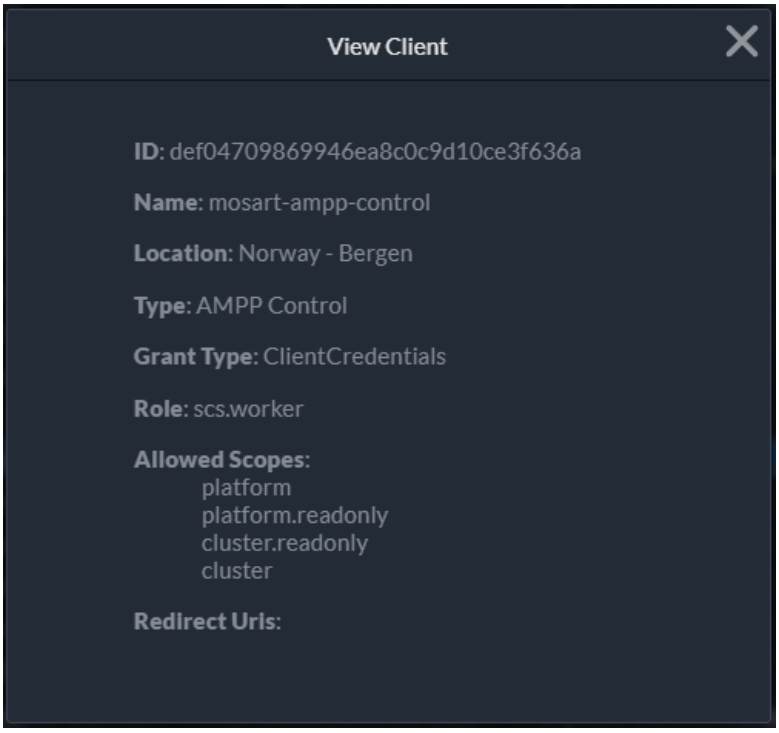
To connect Media Administrator to GV AMPP

To access the GV AMPP platform, you need a specific URL and generated API key, shared as a connection string in Media Administrator.

1. Navigate to the GV AMPP Platform front page by opening **Platform Tools > Identity Management**.
2. On the **Identity Management** page, select **Generate an API Key**.

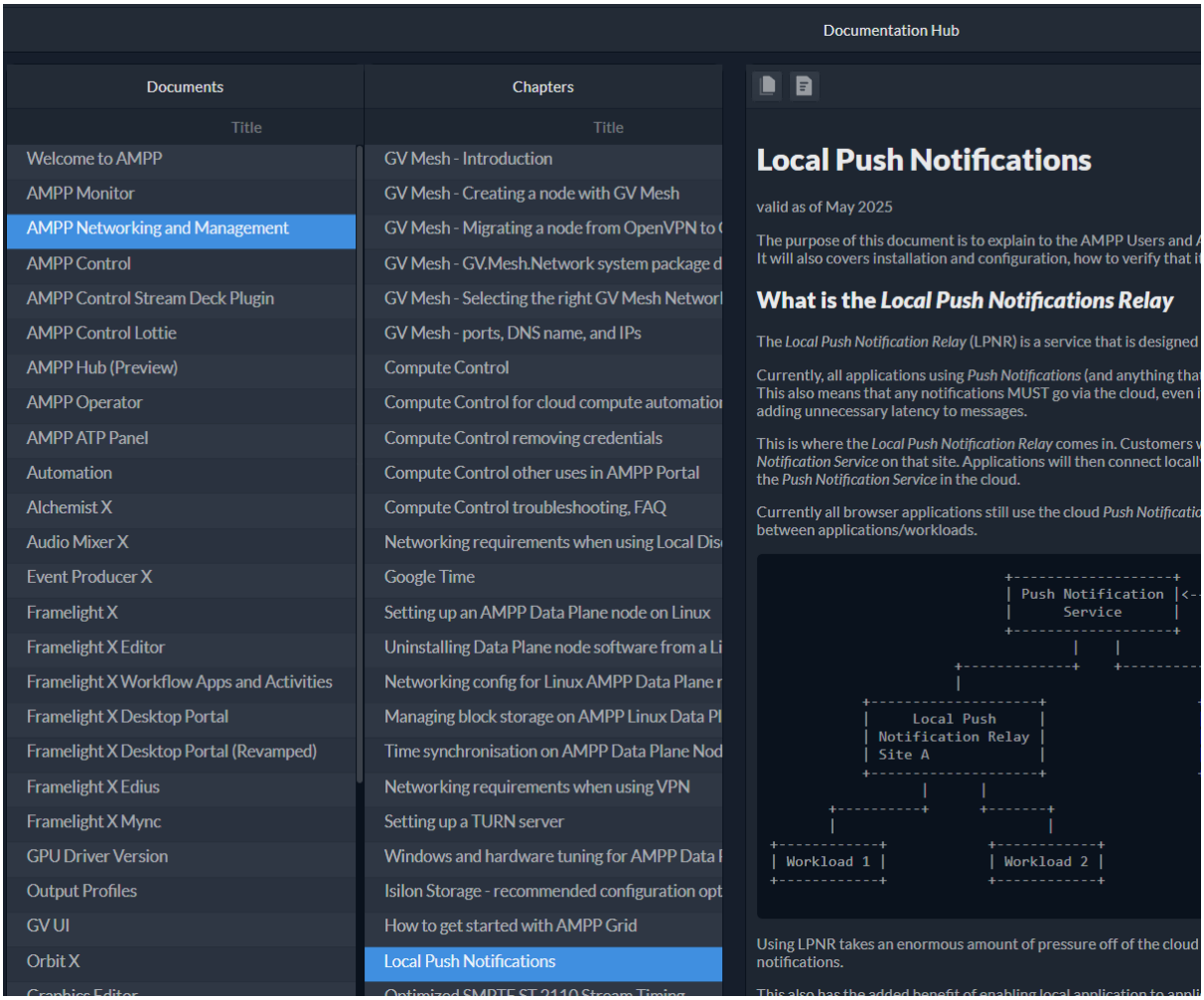


3. Click **AMPP Control**, fill in the details, and *make a note* of the generated key.
4. Update the Role to include `scs.worker`.



This role is needed so Media Administrator can access assets in Framelight X.

5. Once the API key is updated with the necessary permissions, set up the connection to the service:
   a. `Name=gv_ampp;Type=AMPP;PlatformURL=<url>;ApiKey=<api-key>;LoadDatabase=true;`
      (using your `<url>` and `<api-key>` values).
   b. Media Administrator will connect to the service, to load the information about your media.
6. To use the *Local Push Notification Relay* (LPNR) service, you must make an initial setup:
   a. Follow the documentation at the AMPP Documentation Hub (can be reached from the GV AMPP Platform front page).



7. With the LPNR configured, change the connection string to include the `Site Id`:
   a. `Name=gv_ampp;Type=AMPP;PlatformURL=<url>;ApiKey=<api-key>;LocalPushSiteId=<site-id>;LoadDatabase=true;` using your `<url>`, `<api-key>` and `<site-id>` values.
   b. Media Administrator will connect to the service to load the information about your media.

To address distributed site storage

- Write your connection string following the examples below.
  You can identify a specific storage at a named site, by appending keywords *Site* and *Stores*.

Examples and usage

> **Info**:
> - *Site* and *store* names must *not* contain any commas, semicolons, quotes or double quotes.
> Not observing this rule will cause undefined program behavior!
> - You *can* use double-quote marks to enclose *multiple* values for *Stores*. See the example **2** below.

- Any leading or trailing spaces from your naming of either *Stores* or *Site*, are automatically removed.

1. `Site=<name-of-site;Stores=<name-of-store>;` addresses a single site/store.
   To access several sites, add one connection string for each site.
2. `Site=vizrt-hires;Stores="store1, store-2,store 3 ,store#4";` adds multiple stores: *store1*, *store-2*, *store 3* and *store#4*, as a list of names delimited by comma.
   Note that the leading and trailing spaces will be removed for " store-2" and "store 3 " as explained in the Info box above.
3. An example of a connection string that connects to AMPP with two stores named *playback* and *clips*:
   `Site=vizrt-hires;Stores=playback,clips;Name=gv_ampp;Type=AMPP;PlatformURL=https://dev-eu-west.cloud.grassvalley.com;ApiKey=<put-here-your-id>;LoadDatabase=true;`
4. Without keywords keywords *Site* and *Stores*, the entire list of the media is loaded.

To filter assets by type

The connection string can be modified to limit the types of assets Media Administrator fetches and monitors using the keyword *Types*.

- without the keyword or if its value is empty, all asset types will be included
- `Types=video;` returns all videos
- `Types=video, subclip;` returns all videos and sub-clips
- `Types=image;` returns all images

Grass Valley K2 Connection String

Applies to Models

- K2
- Summit

Connection String Properties

- **Type:** GrassValleyK2 (selects the Grass Valley K2 driver for clip handling).
- **Name:** See Name in *Standard Connection String Properties* above.
- **Config**: See Standard Connection String Properties above. The installed xml config file is *ClipServerGrassValleyK2.xml*.
  Unless changed, the value of **Config** should be *ClipServerGrassValleyK2.xml*. See the examples below.
- **Bin:** When LoadDatabase is *true*, a *cache/database* of server clips is built and maintained. Unless IgnoreBin is *true*, this *cache/database* comprises the clips in the bin designated by the two settings Volume and **Bin**.
  Bin must refer to a bin in the volume designated by Volume .
  Default: a string, *default*. (Yes, there is normally a bin named *default)*.
- **Domain**: The domain part of user credentials.
- **DontUsePath**: (Media Administrator only) For each clip and server, Media Administrator keeps track of the 'id' of that clip on that server. By default this id is the *full* path and name of the clip on the server.
  When **DontUsePath** is *true*, the *id* is the clip name only. The default *false* may be used in most cases.

For a *Stratus* system, changing to *true* may be considered. However, this should *not* be combined with setting IgnoreBin = *true*.
Default: *false*

- **EnsureUniqueObjSlug**: This may be left at the default *false* in most circumstances. However, if the *clip_hirespath* from the NRCS is the full clip path, for example, *edl/cmf//BGO-LAB-K2SUMMI/C:/default/Clip*, this should be set to *true*.
  Note that if setting IgnoreBin = *true*, it is not necessary to also set this EnsureUniqueObjSlug.
  Default: *false*.
- **FilterGetXmlChanges:** (Media Administrator only) If *true*, only clip changes involving the bin selected by the properties **Volume** and **Bin** will be handled. Otherwise (*false*), *all* changes are handled. The default *false* may be used in most cases.
  For a *Stratus* system, changing to *true* may be considered. However, this should *not* be combined with setting IgnoreBin = *true*.
- **IgnoreBin**: If *true*, the *Bin* setting (see above) is ignored, and the *cache/database* of server clips (see LoadDatabase) will comprise all clips in the server volume designated by the Volume setting. In this case, the *clip_hirespath* from the NRCS *must* be the full clip path, for example, *edl/cmf//BGO-LAB-K2SUMMI/C:/default/Clip*. Clips in different bins may have the same name, so the path is necessary to identify a particular clip.
  Default: *false*.
- **LoadDatabase**: Whether to maintain a *cache/database* of server clips.
  By default, this *cache/database* comprises the clips in the bin designated by the two settings Volume and Bin. However, if the setting IgnoreBin = *true*, the *cache/database* will comprise all clips in the server volume designated by Volume.
  - For AV Automation, keep the default *false*.
  - For Media Administrator, use *true*.
  - Default: *false*.
- **ModificationsInterval**: (Media Administrator only) Valid only when LoadDatabase is *true.* Interval (in ms) for checking for server changes.
- **Pass:** The password part of the user credentials.
- **Server:** The IP address/hostname of the Grass Valley K2 server.
- **Suite:** This value may be chosen arbitrarily. However, these guidelines must be followed:
  - For each K2 server, two, or even three, different Suite values should be used:
  - First, the values for Media Administrator and AV Automation should be different.
  - Second, in a redundancy setup with Main and Backup Mosart server the two values for Media Administrator should be different.
    However both Main and Backup AV Automation should use the *same* value, different from the two Media Administrator values.
    For further info about *redundancy* setup, refer to Failover with K2 Servers.
- **User:** The username as part of the user credentials.
- **Volume:** When LoadDatabase is *true*, Viz Mosart builds and maintains a *cache/database* of server clips.
  This *cache/database* is limited to clips in the volume designated by this *Volume* setting.
  - If setting IgnoreBin = *true*, the *cache/database* comprises *all* clips in that volume.
  - If IgnoreBin is false, the *cache/database* comprises only the clips in the bin designated by setting *Bin*.
    Bin must refer to a bin in the volume designated by this **Volume** setting.
  - Default: *V:*

Examples

AV Automation:

```
Name=Summit; Type=GrassValleyK2; Server=K2SERVER1;
Config=ClipServerGrassValleyK2.xml; Bin=default; User=GVAdmin; Pass=GVPass;
Volume=V:; Suite=AVA; Domain=K2SERVER1
```

Media Administrator:

You also need to add the connection string in Mosart Media Administrator. This is the same connection string as in AV Automation, except that you need to

- add *LoadDatabase=true* at the end
- use another value for the *Suite* property.

```
Name=Summit; Type=GrassValleyK2; Server=K2SERVER1;
Config=ClipServerGrassValleyK2.xml; Bin=default; User=GVAdmin; Pass=GVPass;
Volume=V:; Suite=GVAdmin; Domain=K2SERVER1;LoadDatabase=true;
```

Configuration File Properties

- **KeepCache**: See General Configuration File Properties above.
- **RefreshOnUpdate**: See General Configuration File Properties above.
- **VerifyClipDelay**: See General Configuration File Properties above.
- **VerifyModifiedClips**: See General Configuration File Properties above.
- **BypassRetrieveServerConfig** (Boolean, default: *false*): (AV Automation only) Whether to *bypass* retrieval of channel configuration from the server. Set to *true* if there are problems connecting to one or more channels (red line).
- **CloseChannels** (Boolean, default *true*): (AV Automation only) Whether channels are closed at disconnect. Use *false* in a redundancy setup with Main and Backup Mosart servers to prevent clip ejection on failover. Otherwise, use *true*. (See Failover with K2 Servers below.)
- **MaxCueRetries**: Maximum number of cue/load attempts. Default: *1*.
- **RecordingBin** (string): (AV Automation only) The server bin where clips are to be recorded. (This bin must be in the *volume* specified by **RecordingVolume**.) Both **RecordingBin** and **RecordingVolume** must be specified if clips are to be recorded in a specific bin. Otherwise, the recording location is undefined.
- **RecordingVolume** (string): (AV Automation only) The server *volume* where clips are to be recorded. Both **RecordingBin** and **RecordingVolume** must be specified if clips are to be recorded in a specific bin. Otherwise, the recording location is undefined.
- **SuspendConnection** (Boolean, default *false*): (AV Automation only) See Failover with K2 Servers below. Use *true* in a redundancy setup with Main and Backup Mosart servers to prevent clip ejection on failover. Otherwise, use *false*.
- **UseExtendedLogging** (Boolean, default *false*): Adds some more logging. Valid only if Verbose logging is turned on. Should normally be kept at the default *false*.
- **UseMovieAddedNotice** (Boolean, default *false*): Whether to utilize the **MovieAddedNotice** element when processing a **MovieAdded** notice.

Copyright © 2026 Vizrt

Info: This setting was introduced to handle added clips at an installation with older GV firmware. In the default *ClipServerGrassValleyK2.xml*, this setting is *true*. (But if the setting is *removed* altogether, the actual *default* value is *false*.) Change to *false* (or remove setting) if added clips are not handled correctly (for example, if a clip added on the GV server does not turn solid blue in the Viz Mosart GUI).

- **UseVideoFormatFromServer** (Boolean, default *true*): Set to *false* if clip durations (for example, as shown in GUI) are incorrect.

Failover with K2 Servers

When GVG K2 servers are connected to both main and backup Mosart servers, any cued or playing clips would be ejected from the K2 ports during failover.
This is not desirable, so a further layer of configuration file properties have been provided for avoiding this:

- **CloseChannels**: (Default: *true*)
- **SuspendConnection**: (Default: *false*).

For a redundancy setup with Main and Backup Mosart servers, use instead these settings:

```
<item name="CloseChannels" value="false" />
<item name="SuspendConnection" value="true" />
```

With these settings, the outgoing Mosart server (e.g. main) will disconnect in a way that allows the incoming Mosart server (e.g. backup) to retake the same connections and channels.

This ensures that no clips are ejected in the process.

The prerequisite for this to work is that the Connection String Properties on main and backup are set up with *identical* **Suite** names (for AV Automation).

Stratus

The two connection string properties

- **DontUsePath**
- **FilterGetXmlChanges**

were introduced at installations with a Stratus system to ensure that deleted clips appeared chequered in the Mosart GUI.

The default value of both settings is *false*, and that has, as far as we know, worked for all other installations. We recommend the following:

1. Start using the default value *false* for both settings.
2. If problems of the type indicated (deleted clips stay solid blue in GUI) are encountered, change both settings to *true*.

Other combinations (than both *false* and both *true*) have not been tested.

Harris Nexio Connection String

The Harris Nexio driver supports Binary-Coded Decimal TCP/IP transport. The driver can be configured to be case sensitive on searches and use Unicode encoding. The Harris Nexio driver supports recording.

- Control via RS-422 can be done through VDCP.

- Maximum five video server units.

Connection String Properties

- **Type:** Harris Nexio
- **Name*:** See **Name** in Standard Connection String Properties above. Defines the ID of this connection.
- **Server*:** Defines the hostname or IP-address of the server.
- **Config:** Defines the name of the configuration file.

Optional

- **Port*:** TCP port to connect to (Default: *557*).
- **Encoding:*** Set to *U* or *Unicode* if it should be Unicode.
- **Case*:** Set to *B* to be case sensitive on clip name searches.
- **ClipNames:*** Property for using exact clip names. If set to *EXACT*, clip names are case-sensitive (Default: *EXACT*).
- **ChangeCheckInterval:** Defines the number of seconds between checks for clip changes.
- **LoadDatabase** (boolean):* Set to *true* to build internal database of clips present on server and enable clip monitoring (Default: *false*).

Info: The asterisk (*) above denotes the configuration file set property can be overridden by a connection string parameter.

Examples

AV Automation:

```
Name=Harris1; Type=HarrisNexio; Server=192.168.1.1; Port=557;
Config=ClipServerHarris.xml
```

Media Administrator:

```
Name=Harris1; Type=HarrisNexio; Server=192.168.1.1; Port=557; LoadDatabase=true;
Encoding=UTF-8; ClipNames=UNEXACT; ChangeCheckInterval=15;
Config=ClipServerHarris.xml
```

Configuration File Properties

- **ChangeCheckInterval** (integer): Time in seconds how often to poll HarrisNexio for clip changes (Default: *5*).
- **Port** (integer): Defines the default connection port (Default: *557*).
- **Encoding** (string): (Default: *ANSI*).
- **Case** (string): (Default: *UPPER*).
- **ClipNames** (string): Property for using exact clip names. If set to *EXACT*, clip names are case-sensitive (Default: *EXACT*).
- **KeepCache** (boolean): When set to *false*, the cache is cleared on reconnects (Default: *false*).
- **RefreshOnUpdate** (boolean): When set to *true*, the cache is updated on each update clip message from the server (Default: *true*).

Copyright © 2026 Vizrt

- **VerifyClipDelay** (integer): Delay in frames to wait before verifying new clips from server (Default: *0*).
- **SupportSearchUsingObjSlug:** Ignores search for objSlug if set to *false* (Default: *true*).
- **SupportSearchUsingRefID:** Ignores search for refID if set to *false* (Default: *false*).
- **MaxCueTime** (integer): Maximum milliseconds to wait for cue (Default: *5000*).
- **ClipNamePattern:** See General Configuration File Properties.

JupiterWebService Connection String

JupiterWebService is an asset management system.

```
Name=myServerName; Type=Jupiter; Url=myConfigFile
```

Example:

```
Name=Jupiter; Type=Jupiter; Url=ClipServerJupiter.xml
```

This server type does not support any additional parameters in the connection string. The following settings must be added in the configuration file.

Configuration File Properties

- **Server** (string): Defines the hostname or IP-address of the server (Default=localhost).
- **Port** (integer): Defines the TCP port to connect to (Default= `8102`).
- **ServiceName** (string): Default=JupiterService.svc.
- **Service** (string): Default=http://\{Server}:{Port}/{ServiceName}.
- **GetStatusCommand** (string): Default={Service}/clipstatus/{0}.
- **GetKeyFramesCommand** (string): Default={Service}/keyframes/{0}.
- **PollIntervalStatus** (string): Default={Service}/keyframes/{0}.

MVCP Connection String

MVCP (Multiport Video Computer Protocol) is a simple request/response protocol, which is implemented over a TCP byte-stream connection (for example, a stream socket).

Applies to Models

- Vizrt Xlator.
- SVT F2C file name conversion service.

Connection String Properties

- **Type:** MVCP
- **Name:** See **Name** in Standard Connection String Properties above. Defines the ID of this connection (Default: *MVCP*).
- **Server:** Defines the hostname or IP-address of the server (Default: *localhost*).
- **Port:** Defines the TCP port to connect to (Default: *5250*).

Configuration File Properties

- **CustomSearch*:** (Example) Set to a unique string like *SVT* to enable a custom search named *SVT* on *objSlug*.
- **SetNameAsRefId*:** Set clip name as ref ID in the media object (Default: *false*)for re-use in other clip servers (Default: *false*).
- **WriteTimeOut:** Sets the amount of time in milliseconds that a write operation blocks waiting for data from the server (Default: *5000*).
- **ReadTimeOut:** Sets the amount of time in milliseconds that a read operation blocks waiting for data to the server (Default: *5000*).
- **AppendExtension*:** A file extension used when sending messages to the MVCP interface (Default: "").
- **RemoveExtension*:** Used to strip the extension of a clip name from the MVCP interface (Default: "").
- **ClipNamePattern:** See General Configuration File Properties.

Info: The asterisk (*) above denotes a custom search called *SVT*, where in this example SVT is set up to process file names with 36 characters. As Nexio only supports 32 characters, *SVT will* expose a mapping proxy with an MVCP interface and a custom command F2C which translates a long name to a shorter one.

Examples

AV Automation:

```
Name=MVCP2; Type=MVCP; Server=192.165.4.1; Port=5250
```

Media Administrator:

```
Name=MVCP2; Type=MVCPSearch; Server=192.165.4.1; Port=5251
```

Omneon Connection String

Applies to Models

- Models with Omneon Spectrum

Note: Viz Mosart is incompatible with Omneon Spectrum versions earlier than 4.6. Additionally, in some cases, clip length is not updated when using Omneon Spectrum 6.4.3 or earlier.
It is recommended to upgrade to Omneon firmware version 7.9.x or later.

Connection String Properties

- **Type:** Connection type: Omneon or OmneonSearch.
- **Name:** See **Name** in Standard Connection String Properties above. ID of this connection.
- **Server:** Hostname or IP-address of the server.
- **Player:** Performs searches associated with directory for the given player if set. If not set, the Directory option should be used to identify the search directory (optional).

- **Directory:** Specifies the search directory (optional).
- **ExtList:** File extensions to be used. List of case sensitive extensions separated by periods.
- **LoadDatabase** (boolean): Set to true to build internal database of clips present on server and enable clip monitoring (Default: *false*).
- **Config:** Name of the configuration file (*ClipServerOmneon.xml*).
- **UseFullPath:** If *true*, the Media Administrator returns full clip paths. The clip ID contains both the Omneon server directory, as well as the clip name. Setting this property to *true* is mandatory if multiple directories are used on a single Omneon server. (Default: *false*).

Examples

AV Automation:

```
Type=Omneon; Name=Omneon; Server=10.211.114.104; ExtList=.mov.MOV.mxf.MXF;
UseFullPath=true; Config=ClipServerOmneon.xml
```

Media Administrator:

```
Type=OmneonSearch; Name=Omneon; Server=10.211.114.104; Player=Play1; clipdir=/fs0/
media; ExtList=.mov.mxf; LoadDatabase=True; Config=ClipServerOmneon.xml
```

Configuration File Properties

- **KeepCache** (boolean): When set to *false*, the cache is cleared on reconnects (Default: *false*).
- **RefreshOnUpdate** (boolean): When set to *true*, the cache is updated on each update clip message from the server (Default: *true*).
- **VerifyClipDelay** (integer): Delay in frames to wait before verifying new clips from server (Default: *0*).
- **DescriptionPattern** (string): Makes it possible to extract a description from the object slug (objSlug) based on this Regex pattern (Default: *[^_]*_(.*)*).
- **DescriptionReplacement** (string): Regex expression referring to first matched group. Used when replacing object slug (objSlug). (Default: *$1*)
- **InvokeServerOnGetClipInfo** (boolean): If *true* (non-zero), a request is sent to the video server when doing clip search. If *true*, all modified clip events for clip not in cache are verified (as if they were created) (Default: *false*).
- **VerifyModifiedClips** (boolean): If *true*, all modified clip events for clip not in cache are verified (as if they were created) (Default: *false*).
- **IgnoreCase:** If *true*, searching for clips is done case insensitive. Must be used in combination with <MatchExpression ignoreCase=*true* /> from General Search Configuration File Properties.
- **GoToTimeCodeTimeOut** (integer): Maximum time to wait (in milliseconds) for successful execution of command to play clip from a given time code (Default: *6000*).
- **UseNativeGoToTimeCode** (boolean): If *true*, it uses GoToTimeCode method available in Omneon API. This may take a longer time to cue a clip at a specified position (Default: *false*).
- **UseStopOnCue:** If *true*, the timeline is stopped, not paused (play with speed 0) when cueing a new clip. This causes cueing to take longer than usual.

- **ClipNamePattern:** See General Configuration File Properties.

  Search Properties: See General Search Configuration File Properties.

Omnibus OPUS Connection String

```
Name=myServerName; Type=OPUS; Url=myUrl; Config=myConfigFile.xml
```

- **Url:** Specifies the URL of the OPUS Interchange web service.

Orad OCIP Connection String

This connection string is used for video servers controlled by OCIP (Open Control Interface Protocol). This includes Orad VJ (obsolete) and Avid FastServe video servers.

```
Type=Orad OCIP; Name=[Name]; Server=[hostname/IP address]; Port=[Port];
LoadDatabase=true; Config=ClipServerOradOcip.xml
```

Example:

```
Type=Orad OCIP; Name=OradVJ; Server=192.168.15.1; Port=10001; LoadDatabase=true;
Config=ClipServerOradOcip.xml
```

Connection String Parameters

- **Type:** Orad OCIP.
- **Server:** Defines the server hostname.
- **Name:** See **Name** in Standard Connection String Properties above.
- **Port:** Defines the server port. By default, the four ports `10001`, `10002`, `10003`, and `10004` are configured on an OCIP-controlled Orad server. We recommend using different ports for AV Automation and Media Administrator.

> **Example:** *10002* for AV Automation and *10003* for Media Administrator.

- **LoadDatabase:** If *true*, the server database is loaded at start-up (Default: *false*). Use *true* for Media Administrator.
- **Config:** Configuration file containing further configuration settings.

Configuration File Parameters

- **AllowSetPosition:** Whether to enable 'scrubbing'/'scrolling' the in-point of a cued clip in the GUI. Default: *false*. Effective only for servers supporting OCIP 3.0.0 or higher. (Only tested with OCIP 4.0.0.)
- **CopyObjSlug:** Whether the value of the clip 'slug' (the slug is the value of the OCIP item parameter set in ObjSlugParameter shall be copied 'upwards' (to Media Admin). Default: *false*.

> **Info**: The value for **CopyObjSlug** in the installed ClipServerOradOcip.xml file, is initially set to *true*. This will be the value for new installations.

If problems are experienced when a clip is first deleted and then re-created with the same SourceId (assuming that is the value of the ObjSlugParameter setting), but a different Id, make sure this setting is *true*.

- **IgnoreGrowingItems:** Whether clips with *growing* parameter set to *true* will be ignored.
  When set, this implies that growing clips are processed as non-existent and are unavailable. Available for Avid FastServe running FSP 2018.8 or higher. The **RequestGrowing** (below) setting must be *true* for this to have effect.
- **LoopPostRoll:** Number of frames before clip-end to restart loop. Set to a positive value to obtain smooth looping.
- **ObjSlugParameter:** The OCIP item parameter to use for clip 'slug'. The *values* of this parameter should match the *values* of the 'clip_hirespath' newsroom tag.
  Default 'Name'. This default value has worked for the video server Orad VJ. Other workflows, especially using Avid FastServe, require using the value 'SourceId'. Other values are not verified as supported.
- **RequestGrowing:** Whether to request the Growing parameter from the server. Default: *false*.
  Set to *true* if the IgnoreGrowingItems feature is to be used.
- **SetInAndOutPoint:** How a clip's in-point and out-point are set. Available values:
  - **FromServer:** (Default) From the item's In and Out parameters. This may yield incorrect results if the In parameter is different from *0*.
  - **ZeroAndDuration:** (Value in the installed ClipServerOradOcip.xml file, so this will be the value for *new* installations.) As *0* and the item's Duration parameter.
  - **Dont:** For future/internal Vizrt Support use. Do *not* use this value.
- **SkipAbortInLoadClip:** If *true*, skip the ChannelAbort command at cue (Default: *false*).
- **TimeoutDefault:** Timeout in milliseconds for commands with no specific timeout set (Default: *1000* which is 1 second).
- **TimeoutChannelAbort:** Timeout in milliseconds for the ChannelAbort command performed at cue.
- **TimeoutChannelJog:** Timeout in milliseconds for the Jog command performed at 'scrubbing'/'scrolling' (See AllowSetPosition).
- **TimeoutChannelPause:** Timeout in milliseconds for the ChannelPause command.
- **TimeoutChannelPlay:** Timeout in milliseconds for the ChannelPlay command performed at cue.
- **TimeoutChannelPlayInit:** Timeout in milliseconds for the ChannelPlayInit command performed at cue (Default: *4000* (4 seconds)).
- **TimeoutChannelSetSpeed:** Timeout in milliseconds for the ChannelSetSpeed command performed at play.
- **TimeoutChannelStatus:** Timeout in milliseconds for the ChannelStatus command.

Copyright © 2026 Vizrt

- **TimeoutCreateCallbackItem:** Timeout in milliseconds for the CreateCallbackItem command at start-up (Default: *2000* (2 seconds)).
- **TimeoutItemCheck:** Timeout in milliseconds for the ItemCheck command occasionally performed at cue.
- **TimeoutItemGet:** Timeout in milliseconds for the ItemGet command occasionally performed at cue, and also for getting status.
- **TimeoutListChannel:** Timeout in milliseconds for the ListChannel command at start-up.
- **TimeoutListItem:** Timeout in milliseconds for the ListItem command at start-up (Default: *2000* (2 seconds)).
- **UpdateInAndOutPoint:** Whether clip in-point and out-point should be copied 'upwards' (to MediaAdmin) for clips with growing duration.
  Default: *false*. Change to *true* if such clips are not shown with correct duration in GUI.
- **UseHashKey:** If *true*, the item Id will be used as clip 'id'. If *false*, clip 'slug' will be used. (See settings ObjSlugParameter and CopyObjSlug above.)
  Default: *false*. However, the value in the installed ClipServerOradOcip.xml file is *true*, so this will be the value for *new* installations.
  If problems are experienced when a clip is first deleted and then re-created with the same SourceId (assuming that is the value of the ObjSlugParameter setting), but a different Id, make sure this setting is *true*.
- **UsePingOcipOperation (boolean):** If *true*, use an OCIP operation for Ping instead of the default TcpIpClient ping (Default: *false*).

Debug Parameters

- **KeepSpeed:** For Vizrt Support use. Default: *false*.
  Do not change!
- **WSACancelBlockingCallExceptionTraceEventType:** The TraceEventType for the WSACancelBlockingCall exception.
  Options: Critical, Error, Warning, Information, Verbose (Default: *Error*).

Applies to Models

- Orad VJ
- Avid FastServe
- Avid Stream IO

Device Prerequisites

- OCIP 1.0.4. Some features require OCIP 3.0.0.
- Avid FastServe. Some features require FSP 2018.8.

Quantel Connection String

Applies to Models

- Connects to Quantel using Quantel CORBA interface.

Device Prerequisites

- Requires Quantel CORBA services to run.

- Serial numbers of playout servers, normally obtained via Quantel Web interface.

Quantel Video Server Connection String

```
Name=myServerName; Type=Quantel; Mode=Player; SerialNo=mySerialNo; IOR=myQuantelIOR;
Slave=mySlaveAddress; Timeout=10; Config=myConfigFile.xml
```

Connection String Properties

- **Name:** See **Name** in Standard Connection String Properties above. A logical name to be associated with the video server. This is used to identify the server across AV Automation and Media Administrator configurations.
- **Type:** Specifies which driver to be used for connecting to the server. This should be set to *Quantel* for both AV Automation and Media Administrator.
- **Mode:** Specifies whether the Quantel connection is used for playing clips or searching (Default: *Search*):
  - **Mode=Player:** The Quantel connection is used for playing clips (AV Automation).
  - **Mode=Search:** The Quantel connection is used for searching (Media Administrator).
- **SerialNo:** Specifies the serial number of the Quantel video server. This number is used to recognize the particular video server within a Quantel zone portal. Serial numbers may be obtained via the Quantel Web interface.
- **IOR:** Specifies the Corba IOR used to establish connection with a Quantel server. This IOR is normally obtained from the system administrator.
- **Slave:** Specifies an optional slave address. Used when Quantel is configured in a redundancy setup.
- **Timeout:** Specifies the timeout in seconds when initially connecting to the Quantel zone portal (ISA manager). Default is *10* seconds.
- **Config:** Specifies a configuration file containing custom configuration for the Quantel server driver. Normally set to *ClipServerQuantel.xml*, which contains the default configuration.

Examples

AV Automation:

`Name=sQ7; Type=Quantel; SerialNo=19343; Mode=Player; IOR=` http://quantel:@192.168.60.33/ZoneManager.ior `; Config=ClipServerQuantel.xml`

Media Administrator:

`Name=sQ7; Type=Quantel; SerialNo=19343; Mode=Search; IOR=` http://quantel:@192.168.60.33/ZoneManager.ior `; Config=ClipServerQuantel.xm`

Quantel Configuration File Properties

- **DefaultTakenPortAction** (enum): Action when not able to take a port when owned by someone else (Default: *Steal*).
  Options:
  - **Steal:** Takes control over the port even if in use by someone else.
  - **Share:** Shares control with the other user. Not recommended.

    - **Cancel:** Cancels the operation. Port is still in control by other user.
- **IssueRequestForTakenPorts** (boolean): If *true*, the user is prompted to confirm taking a port in use by someone else (Default: *true*). A dialog box appears in AV Automation, warning about taking a port that is currently in use. Only applicable when DefaultTakenPortAction=*Steal*.
- **ReleasePortsWithNoAssignedChannels** (boolean): If *true*, any unassigned ports (with no assigned channels) are released on player initiation (Default: *true*).
- **ServerPollFrequency** (integer): Polling frequency for server, in seconds. This is used as heartbeat between Viz Mosart and Quantel. Note that heartbeats are not issued if data has been received from Quantel since last heartbeat. (Default: *30*).
- **ClipNameIsValidNumber** (boolean): If *true* then it is possible to search from Viz Mosart GUI using valid Quantel ClipIDs directly (Default: *false*).
- **QuantelSlugColumn** (string): Specifies the Quantel database column that is used for the clip slug (Default: *Title*).
- **QuantelClipIDColumn** (string): Specifies the Quantel database column that is used for the clip ID (Default: *ClipID*).
- **QuantelSearchColumn** (string): Specifies the Quantel database column that is used for the clip searches (Default: *Title*).
- **MaxPingAttempts** (integer): Maximum ping attempts before a lost connection is detected (Default: *1*). When Viz Mosart detects a lost connection, the current connection is closed followed by attempts to reconnect. For playout servers only (AV Automation).
- **QuantelRefIDColumn** (string): Specifies the Quantel database column that is used for the clip RefID. Used if SupportSearchUsingRefID is set to *true*. (Default: *Title*).
- **SearchUsingServerIdOnly** (boolean): Forces AV Automation to only search for clips using Quantel ClipId's obtained from Media Administrator if *true*.
- **ClipNamePattern:** see General Configuration File Properties.

Quantel Failure Handling Properties

- **QuantelStatusInterval** (integer): Interval in seconds for the Quantel server to send regular status messages. These messages are only sent when the Quantel server is idle. (Default: *2* seconds).
- **QuantelCommErrorNumRetries** (integer): The number of retries in case of a Quantel CORBA operation results in a CORBA_COMM_FAILURE. (Default: *1*).
- **QuantelCommErrorTimeout** (integer): The timeout in milliseconds between retries in case of a Quantel CORBA operation results in a CORBA_COMM_FAILURE. (Default: *100* msec).

Quantel Search Filters

It is possible to define a set of dedicated filters used to either exclude or include hits returned from the Quantel ISA manager. Note that the Quantel ISA manager returns hits for all attached pools. It is necessary to filter out the clips not available on the pools used by the corresponding video servers.

The Quantel search filters are defined in the Quantel configuration file as a list of Filter elements within a SearchFilter element. The following XML structure defines the default filters:

```
<SearchFilters>
    <Filter property="Category" pattern="TEMPORARY CLONE" include="false" />
    <Filter property="PoolID" pattern="[PoolID]" include="true" />
```

```
    <Filter property="IgnoreCase" pattern="true" />
</SearchFilters>
```

Each filter has the following attributes:

- property (string): Name of a Quantel database column. A special property= `IgnoreCase` is available to specify case insensitive or case sensitive searches.
- pattern (string): Regular expression applied on the resulting property.
- include (boolean): Action to take if the regular expression results in a match:
    - include=*false*, the hit is excluded from the search.
    - include=*true*, the hit is returned as part of the search.
    The default example given above shall be interpreted as follows:
- All clips stored in the Quantel database with Category=*TEMPORARY CLONE* are ignored.
- Only clips in the Quantel database with a PoolID that matches the pool identity of the corresponding video servers are returned.
- All searches are case insensitive.

Vizrt TriCaster Configuration

Connection string are used to configure both AV Automation and the Media Admin service. All corresponding parameters can also be provided separately using a configuration string.

Some configuration parameters for both services are common, so they can be stored in a configuration file. This file can then be used by referencing it from the connection string in the form: `Name=TC;Type=TriCaster;Config=ClipServerTriCaster.xml;`.

> Info: In this example *TC* is an arbitrary name, you can use your own. The common approach is keep the configuration in file named `ClipServerTriCaster.xml` and placed into `%ProgramData%\Mosart Medialab\ConfigurationFiles`.

AV Automation configuration

The general format of the configuration string must include `Name` and `Type` fields.

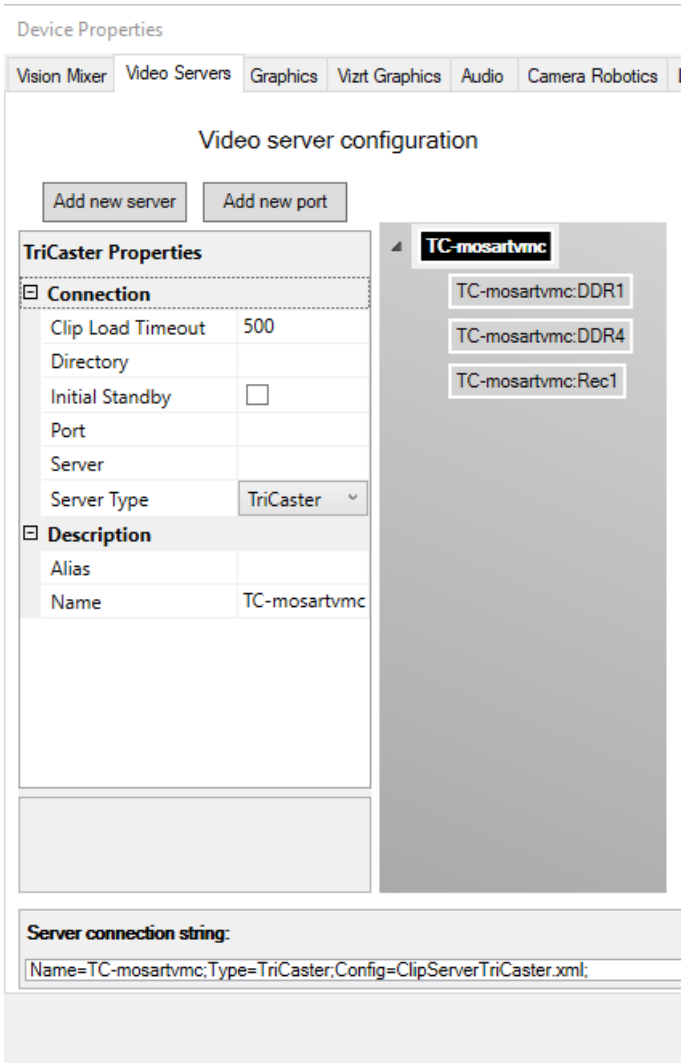
`Name=TC;Type=TriCaster;…`

Then several options are possible: filling out the form, manually entering parameter/value pairs, using a configuration file, or combining these approaches. Use any method is convenient for you, but make sure to check the final configuration string at the bottom of the form.”

Using the configuration file

It helps to share several similar settings between components like AV Automation and Media Admin.

Locate and modify the content of the file ClipServerTriCaster.xml, leaving the form fields empty:

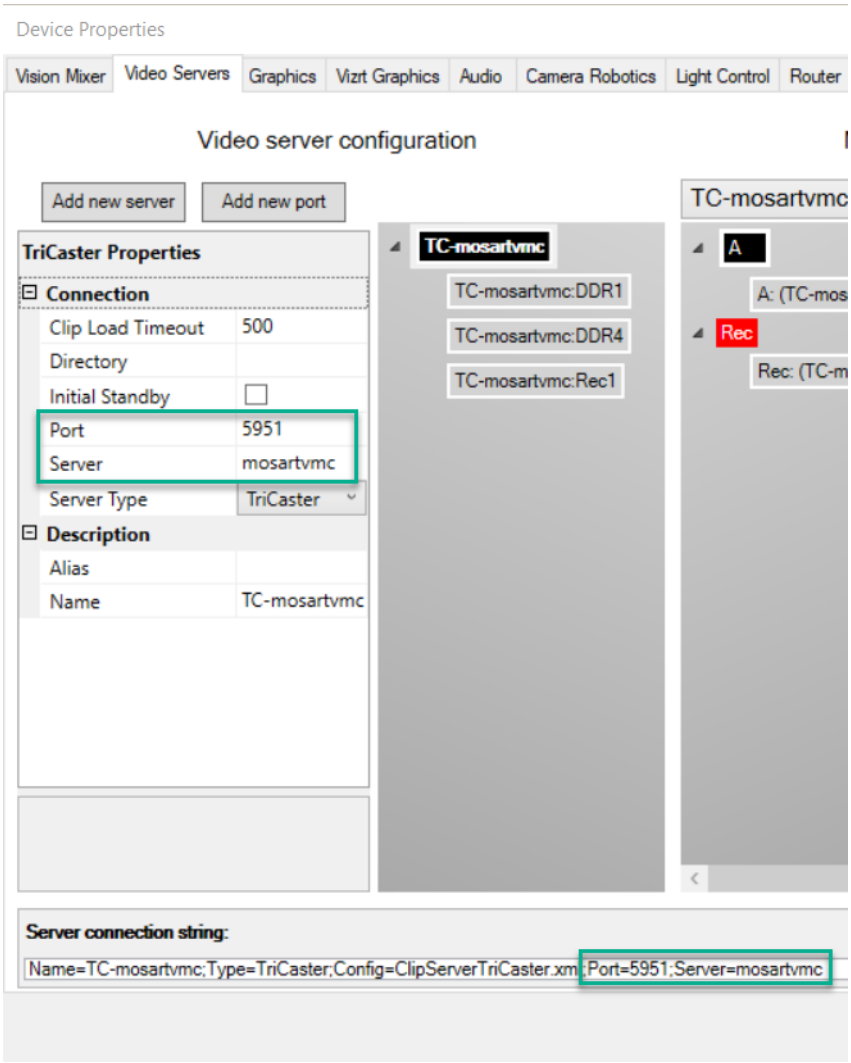


The screenshot above shows the configuration *using the file*, so the corresponding fields in the form are empty. See a description for each parameter below:

- **Server**: Name or IP-address of your TriCaster server.
- **Port**: TriCaster remote control port (usually *5951*).
- **Directory**: The local path on TriCaster used for clips path.
- **ClipLoadTimeout**: How long (in milliseconds) need to wait for the clip loading before setting the start frame, otherwise it will be ignored.
- **HttpUserName/HttpUserPassword/Domain**: Credentials for HTTP authentication.
- **AddExportJobDelayMs**: Delay (in ms) for sending the export job to TriCaster to ensure TriCaster is ready for cutting the story clip from an ongoing recording.

Using properties form

Put the values into corresponding field in the form and check how they were reflected in the connection string in the bottom. Be aware that if you mix the approaches and keep some values in the ClipServerTriCaster.xml its values could be overwritten with entered values.



Entered values like Server and Port put into the connection string and will overwrite the values in ClipServerTriCaster.xml.

Media Administrator configuration

Configuration is performed using a connection string. This string can either include all necessary parameters directly or reference a configuration file containing all required parameters. Referring to a config file can be useful when the file was already created for another component.

The Media Administrator configuration must include two parameters:

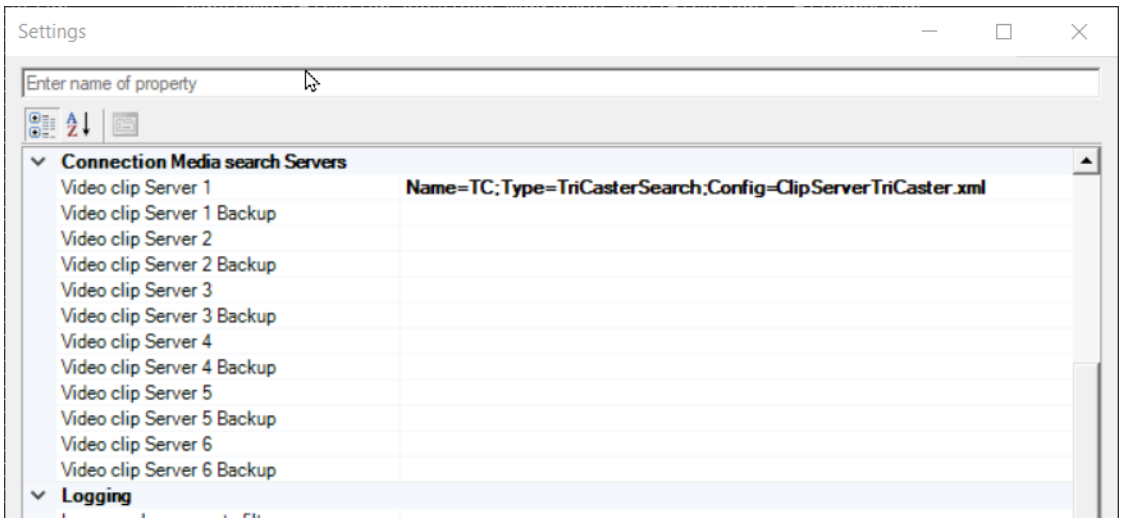
- **Name:** Arbitrary, for example, *TC*.
- **Type**: Must be set to *TriCasterSearch*.

For example, the beginning of the configuration would be:

```
Name=TC;Type=TriCasterSearch;…
```

Place the connection string into any Video Clip Server 1-6 row. The same configuration relates to Backup servers.

The screenshot below shows an example configuration that references an existing, previously created file `ClipServerTriCaster.xml`. This helps when targeting the same video server as used by AV Automation, described above.
If this is not applicable or you need to configure all parameters directly and explicitly, you may skip this example.



If not referring to an existing file, you need to add all parameters and their values, separated with a semicolon.

> **Note**: Parameters explicitly specified in the connection string have higher priority and *override* the corresponding values from the configuration file.

Defaults

The file `ClipServerTriCaster.xml` copied at installation, contains some useful default values, for example, **Port** number is *5951* for each TriCaster.

The configuration starts with the following fragment, which will be refined later:

`Name=TC;Type=TriCasterSearch;Config=ClipServerTriCaster.xml;Server=myTCserver;PollInterval=3;…`

Parameter Descriptions

- **Server**: Name or IP-address of your TriCaster server.
- **Port**: TriCaster remote control port (set by default to *5951*, you can not provide it).
- **HttpUserName**: Username for accessing the TriCaster API.
- **HttpUserPassword**: Password for accessing the TriCaster API.
  Together with **HttpUserName,** as credentials for HTTP/Web Socket authentication.
- **PollInterval**: Interval (in seconds) between polling attempts. Setting to *0* disables polling (not recommended).
  Default: *10* seconds.
- **Folders**: Comma separated list of folders to load clips from.
  If empty, *all* folders will be loaded. Helps with filtering for clips only from specific storages.
  For example, *d:\Media\,d:\Sessions*.

- **InfoBin**: Name of the recorder holding the clip's metadata, for example, *DDR4*.
  TriCaster supports four players, usually named DDR1, DDR2, DDR3, and DDR4.
  When a clip is assigned to a player, all required playback metadata is created here.
  Viz Mosart uses this information to acquire the clip's metadata, by automatically adding the clip to the "bin" of that player. This parameter specifies which of the four players is being used to determine the clip's characteristics.
- **LoadDatabase**: This parameter is used by other systems to determine whether all clips should be loaded into internal cache to minimize answering time. However, TriCaster does not support requests for individual clips, so **LoadDatabase** is ignored, all available clips are always loaded.

Example of complete Media Admin configuration string

```
Name=TC;Type=TriCasterSearch;Server=hosname;HttpUserName=foo;HttpUserPassword=bar;Config=ClipServerTriCaster.xml;Folders=d:\Session1\,d:\SessionN\;PollInterval=100;
```

ScreenLL Connection String

Screen LinguaLib Connection String, used by Media Administrator.

```
Name=myServerName; Type=ScreenLL; Server=myMainServerLocator;
BackupServer=myBackupServerLocator; ServerUsage=myServerUsageMode;
Config=myConfigFile.xml
```

- **Name:** See **Name** in Standard Connection String Properties above.
- **Type=ScreenLL:** Defines the Screen Lingua Lib search driver for clip handling.
- **Server=path:** Defines where subtitle clips are located on the main subtitle server.
- **BackupServer=path:** Defines where subtitle clips are located on the backup subtitle server, if applicable.
- **ServerUsage=mode:** Specifies how availability status is updated based on (main) server and backup server clip status:
  - 0: available if clip is present on main server.
  - 1: available if clip is present on backup server.
  - 2: available if clip is present on either main or backup server.
  - 3: available if clip is present on both main and backup server.
- **Config=ClipServerScreenLL.xml**

SimpleTestPlayer Connection String

The Simple Test Player can be used in Media Administrator for Viz Engine, see the example in section Viz Engine connection string, below.

> **Info**: With the exception of testing purposes, this string provides *no* functionality with AV Automation.

The SimpleTestPlayer may be used in either of two modes:

- **File mode**: A specified file contains clip metadata. The file is read at start-up and is monitored for changes.

- **Folder mode**: A specified folder contains actual clip files. The folder is read at start-up and is monitored for changes.

File mode example:

```
Name=STP;Type=SimpleTestPlayer;File=D:
\DemoClip\DemoClip.xml;Config=ClipServerVizEngine.xml
```

Folder mode example:

```
Name=STP;Type=SimpleTestPlayer;Folder=D:\;Config=ClipServerVizEngine.xml
```

Connection String Properties

Type: SimpleTestPlayer
File: The clip metadata file (used in File mode). See below.
Folder: The clip folder (used in Folder mode)

Clip File

The file must be in this format:

```
<ClipDB>
  <clips>
    <clip slug="..." status="0" dur="..."/>
    ...
  </clips>
</ClipDB>
```

Clip Element Attributes

Attribute	Mandatory	Value	Comment
slug	Yes	String	
status	Yes	0	If missing or other value, the clip is considered missing
dur	Yes	Positive integer	Duration in frames. Should be greater than the value of the MinimumClipLength setting.
objid	No	String	
description	No	String	

Attribute	Mandatory	Value	Comment
mark_in	No	String	
mark_out	No	String	
pdur	No	String	
statusdescription	No	String	
rights	No	String	
refid	No	String	
timestamp	No	String	
type	No	String	
MamId	No	String	

Additionally, the ClipDB element *may* have a *serverDown* attribute. This is for testing purposes only, and should *not* be used.

Configuration File Properties (Folder mode only)

- **InternalBufferSize**: The size (in bytes) of the internal buffer. For details, see Microsoft's article: InternalBufferSize.
- **IncludeSubdirectories**: Whether subdirectories should be monitored
- **MediaInfoMethod**: May have one of these two values:
  - *MediaInfo* (default): Use MediaInfo to retrieve clip file data
  - *VDCPProxy*: In this case, the 'clip files' are regarded as VDCP proxy files. For further details, contact Viz Mosart Support, quoting ~~MOSART-6143~~.
- **SkipExtension**: Whether the file extension is *not* to be part of the clip slug
- **AlwaysGetDuration**: Whether the actual clip duration is retrieved from the clip file
- **UseObjSlugAsServerId**: Whether to use the slug as the server ID. (Otherwise, the path is used.)
- **UseDurationAsMarkOut**: Whether to set InPoint (=0) and OutPoint (=Duration) from Duration

SQL Database Connection String

This connection string (in Media Administrator) is for the SqlMediaSearch driver, for reading clip metadata from an RDBMS (Relational DataBase Management System) using SQL, and then computing clip properties based on:

- SQL query results.
- Existing clip properties (referred to as properties of 'the clip sent down to the SqlMediaSearch driver').

It is assumed that

- All metadata resides in a single database table (the *clip table*).
- The metadata of a single clip is found in a single row (the *clip row*) in this table.

The driver is designed to be used in conjunction with one or more video servers, to provide additional metadata about clips residing on the server(s). The presence of clip metadata in the database is *not* considered proof of clip existence. Only clips that exist on a video server appear as blue in the viz Mosart GUI.

Place this connection string *above* (for example, in Video clip Server 1) that/those of the video server(s).

Applicable Models

The driver has been designed to interact with any RDBMS (Relational Database Management System) using SQL. However, it has only been verified with following versions of Microsoft SQL Server:

- Microsoft SQL Server 2012 (SP1) - 11.0.3000.0 (X64)
- Microsoft SQL Server Express (64-bit) - 11.0.2218.0

Device Prerequisites

Create a DB user with:

- Read access to the table containing clip data.
- No write or admin access. (As no attempts to prevent SQL injection have been done, this is necessary to prevent unauthorized changes to the DB).

General Format

```
Name=DB;Type=SqlMediaSearch;ProviderName=System.Data.SqlClient;Config=ClipServerSql.xml;DbConnectionString="..."
```

Connection String Parameters

- **Name:** See **Name** in Standard Connection String Properties above.
- **Type:** SqlMediaSearch.
- **Config:** See Viz Mosart Video Server Connection String above. A configuration file ClipServerSql.xml is installed. Use the name of this file (*ClipServerSql.xml* unless changed).
- **ProviderName:** The 'invariant name' of a .NET Framework data provider. At least the following providers are installed with the .NET Framework:
  - System.Data.SqlClient
  - System.Data.Odbc
  - System.Data.OleDb
  - System.Data.OracleClient.

  Others *may* have been installed. Only System.Data.SqlClient has been verified.
- **DbConnectionString:** The connection string (containing connection settings) sent to the chosen provider. The format depends on the chosen ProviderName.
  For System.Data.SqlClient (the only verified provider) the format is documented in https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.connectionstring(v=vs.110).aspx.
  Quotes in the value have not been tested and may not work as expected.
- **FileClipProperties:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.

- **FileSelectStatements:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.
- **LogResult:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.
- **DefaultTableName:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.
- **MonitorFoundClips:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.
- **MonitorClipInterval:** Same as the ClipServerSql.xml parameter. As this is set in *ClipServerSql.xml, see below*.

Example

```
Name=DB;Type=SqlMediaSearch;ProviderName=System.Data.SqlClient;Config=ClipServerSql.x
ml;DbConnectionString="Persist Security Info=True;Initial Catalog=HarrisDBTest;Data
Source=BGO-OFRENGINE\SQLEXPRESS;Failover Partner=aut-db-srv03.fs-
pn.vizrtnet.int\harris;User ID=test;Password=test;"
```

Configuration Files

There are *three* configuration files:

- *ClipServerSql.xml*
- *SelectStatements.xml*
- *ClipProperties.xml*

When the Mosart Server is installed, these files are placed in the `ConfigurationFiles` sub-folder of the installation folder. If any file shall be be changed, follow the standard procedure:

- Copy the file to `c:\channeltemplates`, and *change the copy* .
  If you choose to change the name, you must also change (accordingly):
  - If the name of *ClipServerSql.xml* is changed: The Config connection string parameter.
  - If the name of *SelectStatements.xml* is changed: The FileSelectStatements parameter in *ClipServerSql.xml* (below).
  - If the name of *ClipProperties.xml* is changed: The FileClipProperties parameter in *ClipServerSql.xml (below).*

ClipServerSql.xml

This is the main configuration file for the driver. These configuration parameters are available:

- **FileClipProperties:** The name of the file containing clip properties to be pulled from DB. Default: *ClipProperties.xml*, as described below.
- **FileSelectStatements:** The name of the file containing the SELECT statements to be used. Default: *SelectStatements.xml*, as described below.
- **LogResult:** Whether query results should be logged (in Media Administrator console and in Viz Mosart log). Default *true*. Set to *false* if everything works fine, and the messages are superfluous.
- **DefaultTableName:** The name of the outer XML element shown when using LogResult. The default (*dummy*) is usually OK.
- **MonitorFoundClips:** Whether found clips should be monitored. Default *true*. If set to *false*, missing or changed clips are *not* detected.
- **MonitorClipInterval:** If the value is X (and if MonitorFoundClips is *true*), each found clip is re-queried every X seconds. Increase if DB traffic is too high.

- **DbConnectionString:** Same as the connection string parameter; As this is set in the connection string parameters, see above.
- **ProviderName:** Same as the connection string parameter; see above. Should be set in the connection string.

In addition, these 'common'/'standard' parameters are available:

- **SupportSearchUsingObjSlug:** See General Configuration File Properties above.
- **SupportSearchUsingRefID:** See General Configuration File Properties above.

SelectStatements.xml

If the name of this file is changed, the configuration parameter FileSelectStatements in ClipServerSql.xml *must* be changed accordingly.

To further understand this configuration, please study the examples provided in the installed file.

This file defines the SQL SELECT statements that are executed when:

- The DB is queried for data on a single clip.
- The DB is searched for clips satisfying a given search criterion.

Each of these two SQL SELECT statements is defined by a SelectStatement element. For technical reasons, the two SelectStatement elements are enclosed in a Statements element, which in turn is enclosed in a SelectStatements element.

A **SelectStatement** element has these attributes:

Attribute	Value(s)	Description
type	Single/Multiple	Distinguishing between the two types of SQL SELECT statements.
database		The DB name.
schema		The schema name.
table		The table name.

A **SelectStatement** element has these sub-elements:

Sub-element	Description
SelectColumns	The columns in the SELECT clause of the SQL statement.
WhereColumn	The WHERE clause column to be searched for the (variable) search criterion.
AdditionalWhereColumns	Additional WHERE clause columns being searched for *constant* values.

The **SelectColumns** element has one or more SelectColumn sub-elements, each with these attributes (and no sub-elements):

Attribute	Value(s)	Description
name		The DB column name
type	String/Int	The DB column data type. For Microsoft SQL Server, • String is used for char and varchar columns. • Int is used for int columns.
format	Default/ HhMmSsFfPackedBCD	How a column value is to be interpreted: • Default: The value is passed as is. (And this *is* default, so it is used if no format attribute is present.) • HhMmSsFfPackedBCD: The (four-byte) int value is interpreted as the packed BCD representation of a time code hh:mm:ss:ff. This is converted to the total number of frames. E.g., the int (decimal) 404310278 is hex 18194906, corresponding to the time code 18:19:49:06, in total 1649731 frames.

The **AdditionalWhereColumns** element has one or more **WhereColumn** sub-elements.

A WhereColumn element (either as a direct sub-element of SelectStatement, or as a sub-element of AdditionalWhereColumns) has these attributes:

Attribute	Value(s)	Description	Comment
name		The DB column name	
type	String/Int	The DB column data type. For Microsoft SQL Server, • String is used for char and varchar columns. • Int is used for int columns.	Only *String* has been tested.
value		The value to search for	Used only for WhereColumn elements which are sub-elements of AdditionalWhereColumns. For the WhereColumn sub-element of SelectStatement, the value to search for is given by context.

Attribute	Value(s)	Description	Comment
searchMethod	BeginsWith/ IsEqualTo/ EndsWith/ Contains	How the column value (c) is compared to the value to search for (s): • IsEqualTo: c must be exactly equal to s • BeginsWith: c must begin with s • EndsWith: c must end with s • Contains: c must contain s	Only IsEqualTo makes sense for Int. EndsWith and Contains have *not* been tested. IsEqualTo has not been tested for Int.

ClipProperties.xml

If the name of this file is changed, the configuration parameter FileClipProperties in ClipServerSql.xml *must* be changed accordingly.

To further understand this configuration, please study the examples provided in the installed file.

This file defines the clip properties computed by the driver. Each of these properties is defined by a Property element. For technical reasons, the Property elements are enclosed in a Properties element, which in turn is enclosed in a ClipProperties element.

A **Property** element has one attribute:

Attribute	Description	Comment
name	The property name	Only these have been implemented: • Description (the clip description as shown in the GUI (if selected by the setting Video clips should be presented with there)) • InPoint (the clip in point as shown in the GUI Preview window) • ObjSlug (the clip slug as shown in the GUI Media pool Search tab) • OutPoint (the clip out point as shown in the GUI Preview window)

A **Property** element has one sub-element, an *expression*. The principal feature of an expression is that it may be evaluated to yield a value. There are four kinds of expression elements (so a Property element has one of these as a sub-element):

Sub-element	Description	Comment
ClipExpression	The value of a ClipExpression depends only on a property of the clip as sent down to the SqlMediaSearch driver.	The clip properties available depend on the video server driver(s) used together with SqlMediaSearch. Only the BasePoint property (start timecode in frames, available for the HarrisNexio driver) has been tested.

Sub-element	Description	Comment
DbExpression	The value of a DbExpression depends only on the value of a column in the clip row (the DB row pertaining to the clip).	The column has to be defined by a SelectColumn element as described in the **SelectStatements.xml** section above.
ConstantExpression	The value of a ConstantExpression is (as strongly indicated by the name) constant , it depends neither on the clip as sent down to the SqlMediaSearch driver, nor on the DB.	ConstantExpressions were introduced for internal purposes, and have not been real-life tested.
OperationExpression	An OperationExpression is used to combine other (simpler) expressions by performing an operation, which means applying an *operator* to the values of its *operands* (sub-expressions). Each operand/sub-expression may be of any of the four kinds of expression listed in this table. In particular, OperationExpressions may be nested to any depth.	

A **ConstantExpression** element has one attribute:

Attribute	Description
value	The (constant) value of the ConstantExpression

Both a **ClipExpression** and a **DbExpression** has two attributes:

Attribute	Description
name	See table below
default	The value to be used if evaluating the expression as described in the table below doesn't make sense in the given context

The **name** attribute and expression evaluation is described in more detail for each of the two kinds of expressions:

Expression kind	Description of the name attribute	Expression value	Comment
ClipExpression	The name of a clip property	The value of the property for the given clip	Only the BasePoint property (start timecode in frames, available for the HarrisNexio driver) has been tested.

Expression kind	Description of the name attribute	Expression value	Comment
DbExpression	The name of a column in the DB clip table	The value of the column for the clip row	The column has to be defined by a SelectColumn element as described in the **SelectStatements.xml** section above.

An **OperationExpression** element has one attribute:

Attribute	Value(s)	Description	Comment
operator	Minus/Plus	The operator to be applied to the values of the operands. Details are given in the table below.	Both operators make sense for numerical arguments only, and have been implemented for integer arguments only.

An **OperationExpression** element has one sub-element: Operands. An Operands element has no attributes, but any number of expression sub-elements, meaning any number of ClipExpression, DbExpression, ConstantExpression, and OperationExpression sub-elements.

These sub-elements are the operands / sub-expressions of the OperationExpression.

Some further details on each operator:

Operator	Expression value
Minus	If there is one operand, the negation of the value of that operand. If there are two (or more) operands, the difference between the values of the first operand (minuend) and the second operand (subtrahend).
Plus	For any number of operands, the sum of the values of the operands. (As a special case, the 'sum' of no operands is zero. However, the Operands sub-element itself must be present. so <ConstantExpression value= `0` /> is convenient.)

VDCP Connection String

The VDCP driver supports both serial and TCP as transport. Video Disk Communications Protocol (VDCP) is also known as Louth protocol. The driver can be configured to run in eight character or variable character mode. (See the UseLongClipNames setting below.)

> **Note**: Standard VDCP does not support *looping* of clips. For a very limited number of servers, a customized solution has been introduced to support looping over VDCP, if described in the server model's configurations.

Applies to Models

- Dalet BRIO

- Dayang VIPS-2H-4XHD
- EVS XT-VIA
- Ross Blackstorm
- Vector MultiPlay VServer

Device Prerequisites - VDCP

- Dalet BRIO
  - Single VDCP connection configuration per port - does not support multiple control ports per connection.
  - Only clips in folders added to the BRIO VDCP configuration are visible for Viz Mosart to monitor and play-out.
  - Duration of video files is only available if the VDCP connection is configured to use a physical play-out port.
- Dayang VIPS-2H-4XHD
  - For Media Administrator to function correctly, the Dayang server must be configured so that a dedicated signal port is available for media search. Viz Mosart Support does not have details about this procedure and recommends consulting Dayang.

Connection String Properties - VDCP

- **Type:** VDCP (serial) or *VDCPtcp* (Ethernet).
  For Dayang, only VDCP (serial) has been tested.
- **Name:** See **Name** in Standard Connection String Properties above.
- **Server:** Hostname or IP-address of the server (for Type=*VDCPtcp* only).
- **Port:**
  - Serial port (Type=VDCP, Default: *COM1*)
  - Server side TCP port to connect to (Type=VDCPtcp, Default: *5205*).
- **LoadDatabase:** If *true*, builds internal database of clips present on server and enables clip monitoring (Default: *false*).
- **Config:** Name of the configuration file. At installation the file `ClipServerVDCP.xml` is copied to the ConfigurationFiles subfolder of the installation folder. Unless you *rename* this file, use Config=*ClipServerVDCP.xml*. (You may need to copy and rename if you have several VDCP servers which should be configured differently.) This file may be taken as a starting point for custom configuration, See Configuration File Properties - VDCP below.
- **DisableHeartbeat (boolean):** Whether the 'Ping' operation (see NextPingDelay above) is just a simple connection check. If *false*, the Ping operation sends a command to the server. For some servers, this is necessary to prevent the server from closing the connection. This applies to at least the Ross Blackstorm server.
- **NoOfPlayers**: The number of signal ports on the server to be made available to the program (AV Automation or Media Administrator). (The ports are numbered consecutively from 1 upwards.) Default 5. (So then ports 1-5 are available.) Use a larger value if you need to control a signal port with a number greater than 5.
- (MediaAdministrator and Dayang only) **PlayerToOpen**: The number of the dedicated signal port available for media search. If this number is greater than 5, the NoOfPlayers must be set to at least this number.

Note: There have been known issues with this vendor's implementation of the VDCP protocol. Vizrt recommends that you consult Dayang if there are issues where the video server seems to require an open port to respond to ID List command.

Examples

AV Automation: **VDCP (serial)**

```
Type=VDCP; Name=Server1; Port=COM1; Config=ClipServerVDCP.xml
```

AV Automation: **VDCPtcp (Ethernet)**

```
Type=VDCPtcp; Name=Server1; Server=192.168.42.42; Port=10001;
Config=ClipServerVDCP.xml
```

AV Automation: **Dayang**

```
Name=DY1;Type=VDCP;Port=COM4
```

Please see the Note below.

Media Administrator: **VDCP (serial)**

```
Type=VDCP; Name=Server1; Port=COM2; LoadDatabase=true; Config=ClipServerVDCP.xml
```

Media Administrator: **VDCPtcp (Ethernet)**

```
Type=VDCPtcp; Name=Server1; Server=192.168.42.42; Port=10002; LoadDatabase=true;
Config=ClipServerVDCP.xml
```

Media Administrator: **Dayang**

```
Name=DY1;Type=VDCP;Port=COM3;Config=ClipServerVDCP.xml;LoadDatabase=True;NoOfPlayers=
101;PlayerToOpen=101
```

Note: For Dayang, different COM ports should be used for AV Automation and Media Administrator

Configuration File Properties - VDCP

- **DefaultRecordingLength (integer):** Default duration of recorded video files in frames *(Default:1000).*
- **DelayGetClipDuration (Boolean):** Enable to build the initial playlist without getting the file duration (Default: f*alse*).

- **IgnoreClipExistence (Boolean):** Enable to ignore requesting clip presence on the server. This should be set to *false* in the configuration file used by AV Automation (Default: *true*).
- **LogMessages (Boolean):** If *true*, log any messages that are sent and received. Verbose logging must be activated for this to have effect (Default: *false*).
- **MaxClipList (integer):** The maximum amount of clips to hold in cache. When this limit is reached, there are no more clips added to the cache. Setting this to a low value reduces load time, but might cause invalid clip status if non-cached clips are in the rundown.
- **ClipListChangesInterval:** The interval, in seconds, to poll for clip list changes (added/deleted clips) on the Media Admin server (Default=*30*). See also UseAddedClipsCommand.
  Requires LoadDatabase=*true* (set in connection string) and UseAddedClipsCommand=*true*.
- **DelayUpdateClipsMsec:** Delay, in milliseconds when to trigger a clip update after retrieval of clips. Used when obtaining multiple clips when loading and updating the database (Default=*5000*).
  Requires LoadDatabase=*true* (set in connection string) and DelayGetClipDuration=*false*.
- **DelayLoadDatabaseMsec:** Delay in milliseconds for delaying loading the database after connecting to the server (Default=*5000*).
  Requires LoadDatabase=true (set in connection string).
- **RequestPortStatus123 (Boolean):** Whether to request more status from the VDCP server. May safely be left at *false* for normal production. Change to *true* for debugging purposes if advised to by Vizrt Support.
- **SendStillWhenCueDone (Boolean):** When the server has cued a clip, whether to send a command making sure that the first frame is shown on the server output. Change to *true* if supposedly cued clips are not shown on the server output.
- **SendStopBeforeCueing:** Whether a Stop command is always sent before cueing a clip. Change to *true* if cueing problems are encountered, in particular when cueing/playing adlibs back-to-back (Default=*false*).
- **StopShouldCheckForIdle (Boolean):** Whether stopping a clip should make sure that the port is idle. Change to *true* if cueing problems are encountered.
- **TimeoutPlay (integer)**: The time (in ms) for the server to acknowledge that it is playing the cued clip. Used only if VerifyPlay is *true* (or if EnableLoopingDaletBrio is *true*, and the cued clip should loop).
- **UseAddedClipsCommand (Boolean):** Enable to retrieve a list of new files added to the video server.
- **UseFixedPortStatusBitMap (Boolean):** Whether to force Mosart to use a fixed bit map when requesting port status. Change to *true* for new installations. In general, CHANGE the value if problems are encountered, in particular problems related to cueing.
- **UseInternalPlayerState (Boolean):** Enable to ignore requesting presence of the clip on the play-out device using the ID REQUEST message.
- **UseLongClipNames (Boolean):** If *true*, use messages supporting variable length file names (maximum length 80 characters). If *false*, use fixed eight character files names padded with spaces if less than eight characters.
- **VerifyPlay (Boolean):** Whether to verify that the server is actually playing.
- **WaitTimeoutPlay (Boolean):** The time (in ms) to wait between each server playing status polling. Used only if VerifyPlay is *true* (or if EnableLoopingDaletBrio is *true*, and the cued clip should loop).

Configuration File Properties - Dalet BRIO only

The following properties are used for the looping functionality of the Dalet BRIO:

- **EnableLoopingDaletBrio (Boolean):** Whether to enable the looping mechanism for Dalet Brio. Set to *true* only if this file is for a Dalet BRIO, and you want looping (Default: *false)*.

- **TimeoutDaletBrioLoop (Boolean):** The time (in ms) to wait for the server to acknowledge that looping has been turned on/off.
- **WaitTimeoutDaletBrioLoop (Boolean):** The time (in ms) to wait between each server looping status polling.

Viz Engine Connection String

Viz Engine Connection String, used by Media Administrator and AV Automation.

Connection String Properties

- **Type:**
  For AV Automation: *VizEngine*.
  For Media Administrator: *SimpleTestPlayer*.
- **Name:** See **Name** in *Standard Connection String Properties* above. Defines the ID of this connection.
- **Server:** Defines the hostname or IP-address of the server (for Type=*VizEngine* only).
- **Port:** Defines the port.
- **Config:** Defines the name of the configuration file.
- **Folder:** Location of the Viz Mosart configuration files. Either in `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\ConfigurationFiles` or with the video configurations in `C:\channeltemplates`.

Examples

Connection string for Integrated Engine (Viz Engine connection in AV Automation):

`Type=VizEngine;Server=localhost;Config=VizEngineSwitcherConfig.xml (Port=6100 is optional)`

Connection string for Video Server (Viz Engine connection in AV Automation):

`Name=Vizrt;Type=VizEngine;Server=localhost;Port=6100;Config=ClipServerVizEngine.xml`

Connection string for Media Administrator:

`Name=VizEngine;Type=SimpleTestPlayer;Folder=D:\;SkipExtension=true;Config=ClipServerVizEngine.xml`

Configuration File Properties

- **AlwaysGetDuration** (Default: *false*):
  - When *false*, only files in the active Viz Mosart rundown get the actual video duration.
  - When *true*, get the video duration from all files in the monitored folder. Note that setting this value to *true* impacts the startup time of the SimpleTestPlayer driver in Media Administrator for folders with a large number of video files.

Viz One and Media Service Connection String

This connection string is used by Media Administrator for Viz One and Media Service.

Viz Mosart can search for clips in Viz One and initiate transfer of Viz One clips to a specified playout storage. The driver uses the REST-based Viz One Third party API to execute tasks.

> **Info**: If Viz One is configured with HTTPS, make sure the *Viz One server certificate is* ***trusted on the Mosart*** *machine*.
> For example, if Viz One uses an *internal/self-signed certificate*, install the root CA certificate into the Windows **Trusted Root Certification Authorities** store on the Mosart machine.
> When configuring the connection string in Media Administrator, use the **exact hostname or IP address that the certificate was issued for**.

Device Prerequisites (Viz One)

For further details, please consult the *Viz One Administrator Guide*.

- At least one Viz One *user* at least one *publishing point* must be configured.

Connection String Properties

Naming	Examples	Description
Name	VizOne VizEngine MediaSearch	Name of connection. Must be same as Name of the video server in AV Automation. See **Name** in Standard Connection String Properties above.
Type	VizOne	Type of connection. Must be *VizOne* to use the Viz One Clip Administrators.
UserName	vizmse	Viz One only Username for user that connects to third party API. Please look at separate section in this document on how to create a dedicated user for Viz Mosart to access Viz One. *admin* as username and password may be used for testing, but should *not* be set in production!
Password	vizmse	Viz One only Password for user that connects to third party API

Naming	Examples	Description
Https	true false	If Viz One forces HTTPS (HTTP not allowed), set *Https=true* in the connection string to communicate with Viz One using **HTTPS** for all operations (authentication, clip updates, publishing, etc.). While HTTP may work if Viz One is set up to allow it, this is insecure and not recommended. If Viz One is configured for HTTPS, Mosart should also be configured with Https=true, and the certificate must be trusted on the Mosart machine. *Default: false.*
Server	10.211.112.213 bgoviz1 bgoviz1.media.com	IP address, host name or the fully qualified domain name (FQDN) of Viz One server. If https is setup, use the **exact hostname/IP address/FQDN that the certificate was issued for**.
PublishingPoints	vve2-1 vizengine-1 MediaService	A publishing point is a playout storage defined in Viz One. This can be accessed through Administration menu on the Web interface. Comma separated list of publishing point identifiers where the clip administrator should ensure assets are present. If *MediaService* is specified for PublishingPoints, Media Service and not Viz One is used to search for clips.
ValidDuration	26	Viz One only Duration for which an asset should be present on the publishing point from time of transfer. **Value is in hours**. Default: 24
AutoTransfer	true false	Viz One only *true*: Any assets in the playlist that is present in Viz One, but not on the publishing points will be transferred. *false*: Assets are marked as not present. Default: *true*.
PublishingPointAsObjId	true false	*true*: An ObjectId will be set for each publishing point as well as Name from connection string (an additional server ID is set for each publishing point). *false*: Use Name from connection string as ObjId (only one server ID is set: for Name). Default: *false*.
Port	21099	Media Service only MediaService requires port to be specified. Default port is *21099* for MediaService. See also *Configure Media Service* section in the Media Service guide. (Viz One does not require this field).

Naming	Examples	Description
Config	ClipServerVizOne.xml	Name of configuration file located in ChannelTemplates folder. See also Standard Video Server Connection String.
GetMetaGraphics	true false	When *true*, Mosart automatically fetches overlay graphics (metagraphics) from Viz One and attaches them to the clip in the rundown. When *false*, any metagraphics embedded in the clip are ignored by Mosart. Default*: true.* **When to use *true*:** • In workflows where Viz One is the **single source of truth** for clips and their associated graphics. • Suitable when customers want to treat clips as Viz One *items* (data elements in the Viz One database). Graphics can be reused and edited from one place, so Mosart should always fetch them. **When to use *false*:** • In workflows where the NRCS or Viz Pilot Edge is responsible for attaching graphics to clips. • Suitable when customers want to treat clips more like Viz Mosart *templates*, where new versions are created or edited independently. This avoids Mosart duplicating graphics by fetching them again from Viz One.

Example (Media Service)

Name=VizEngine;Type=VizOne;Server=localhost;Port=21099;PublishingPoints=MEDIASERVICE;Config=ClipServerVizOne.xml

Example (Viz One)

Name=VizOne;Type=VizOne;UserName=username;Password=password;Server=10.211.112.213;PublishingPoints=vizengine-1;ValidDuration=26;AutoTransfer=true;PublishingPointAsObjId=false;Config=ClipServerVizOne.xml

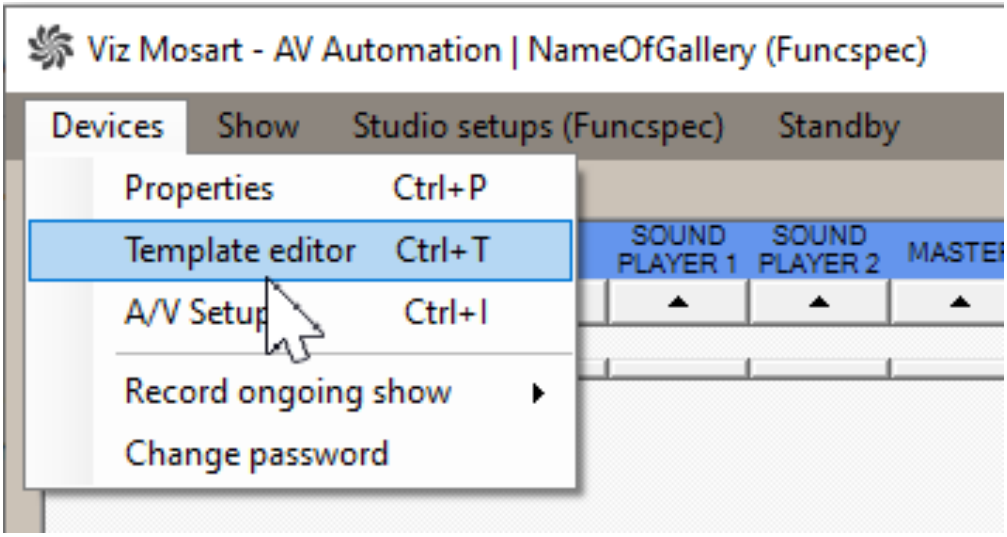
Info: The differences between *ClipServerVizEngine.xml* and *ClipServerVizOne.xml:*

- **ClipServerVizEngine.xml**: Handles video server in configuration AV Automation
- **ClipServerVizOne.xml**: Only for establishing clip searches in Media Administrator

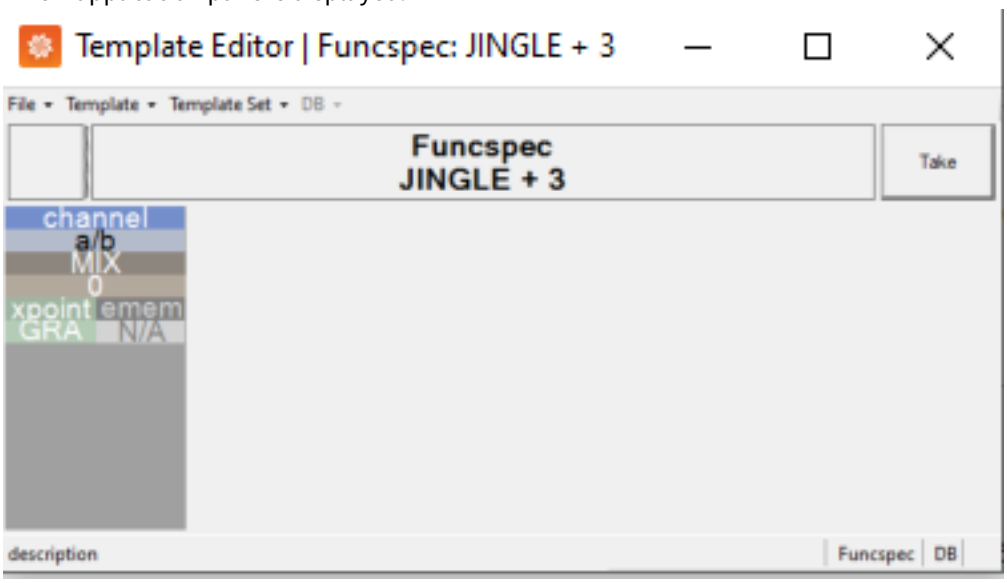
5.7 Template Editor

You create templates as part of your show design. Use the **Template Editor** to configure the Viz Mosart templates.

- From **AV Automation**, select **Devices** > **Template editor**.



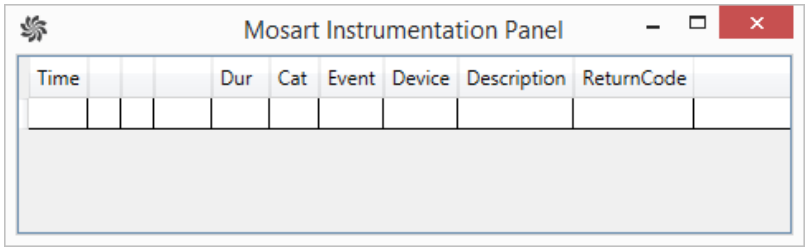
A new application pane is displayed:



- From this view you can select the required template/template variant (with **Template > Select**) and work with the device settings as described in Template Device Functions.

Show design and detailed template topics are presented in section Show Design and Template Editor.

5.8 Mosart Instrumentation Panel



The **Mosart Instrumentation Panel** displays all device commands in real time, that are being sent to the various connected devices. Each command is time stamped, revealing when a device command was issued, in relation to the taking of a Mosart template.

5.9 Recording a Show

Viz Mosart can instruct a connected video server to initiate recording.

> **Note**: The **Story Recorder** feature of Viz Mosart offers more versatile record-pause-retake features, beyond the recording functionality described in this topic.
> Please refer to section Story Recorder Mode.

5.9.1 Working with Recordings

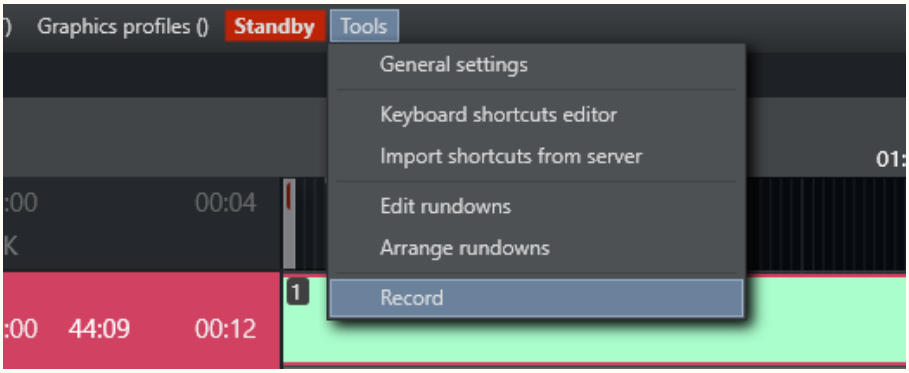
- Prerequisites
- Managing Recordings

Prerequisites

The video server must first be configured, as explained in the procedure **To add a recording port** of Video Servers.

- The configured video servers are displayed in AV Automation under **Devices > Record**.

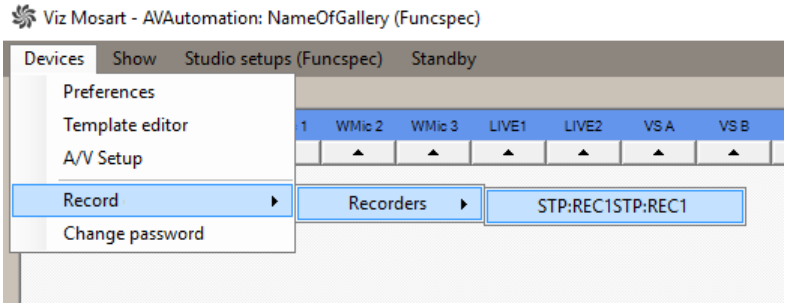
> **Note:** Recording is also presented in the Viz Mosart UI under **Tools > Record**



Managing Recordings

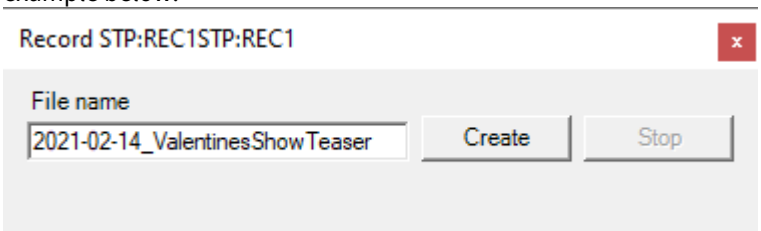
To Start a Recording

1. In AV Automation, select **Devices > Record > Recorders > [Name of recording port]**

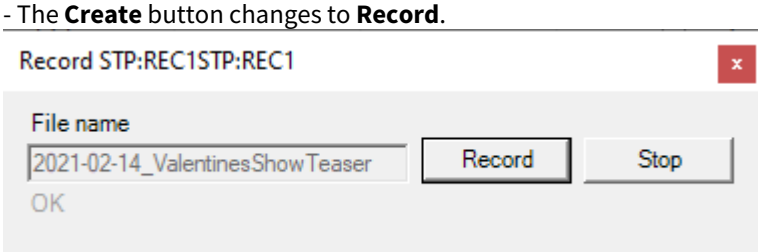


Note: If several recording ports have been configured, the Recorders menu will have several menu items.

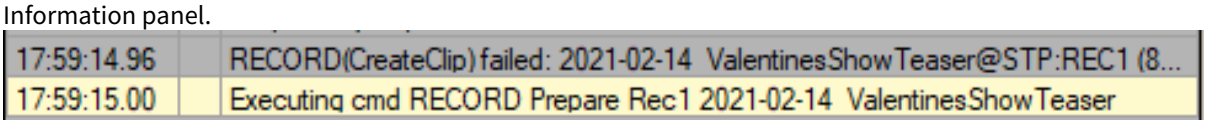
2. In the dialog box, enter a file name that will make it simpler to identify your recording, as shown in the example below:



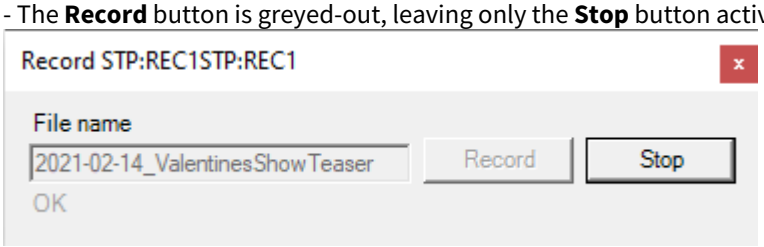
3. Click **Create**.
   The appearance of the dialog box changes.
   - The **Create** button changes to **Record**.



   - The commands that are sent in the background to the video server, are presented in AV Automation's Information panel.



4. Click **Record**.
   The appearance of the dialog box changes once again.
   - The **Record** button is greyed-out, leaving only the **Stop** button active.



   - The commands that are sent in the background to the video server, are presented in AV Automation's Information panel.

18:00:38.18		RECORD(RecordChannel) failed: @STP:REC1 (2 ms): 403 Unit/port STP:REC1...
18:00:38.18		Executing cmd RECORD Start Rec1 2021-02-14 ValentinesShowTeaser

Recordings are stored on the video server's system. Exactly where, is dependent on the video server type.

To Stop a Recording

To stop an ongoing recording,

- Click **Stop**.
  The commands that are sent in the background to the video server, are presented in AV Automation's Information panel.

18:04:07.02		Executing cmd RECORD Stop Rec1 2021-02-14ValentinesShowTeaser

5.9.2 Supported Video Servers

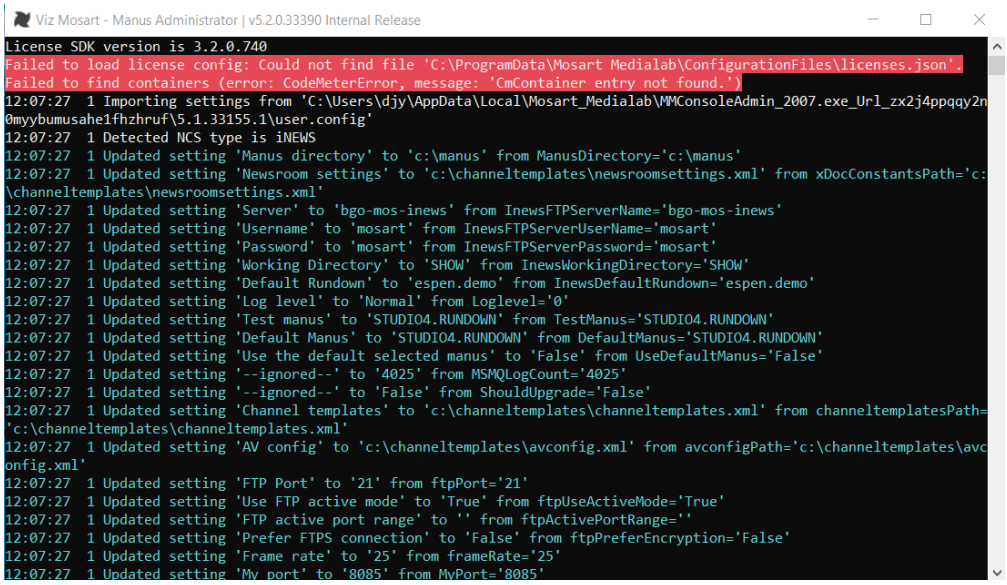
Viz Mosart is continually improved to extend the range of video server drivers that are supported for recording.

Driver	Supported since
Omneon	2010-01
Harris Nexio	2011-09
VDCP	2014-07
Quantel	2016-08
MVCP	2016-08
Grass Vally K2	2016-09
AirSpeed MultiStream	2018-05

Copyright © 2026 Vizrt

6 Manus Administrator

Manus Administrator is the intelligent element of Viz Mosart, monitoring the rundown in the connected NRCS, parsing the content into Viz Mosart functionality that is presented in the Viz Mosart GUI.



Manus Administrator is a console application and must be kept running at all times on your Mosart server.

6.1 NRCS Software Compatibility

Info: From Viz Mosart 5.13, the legacy Manus Administrator applications for MOS and iNEWS are no longer included in the installation software.

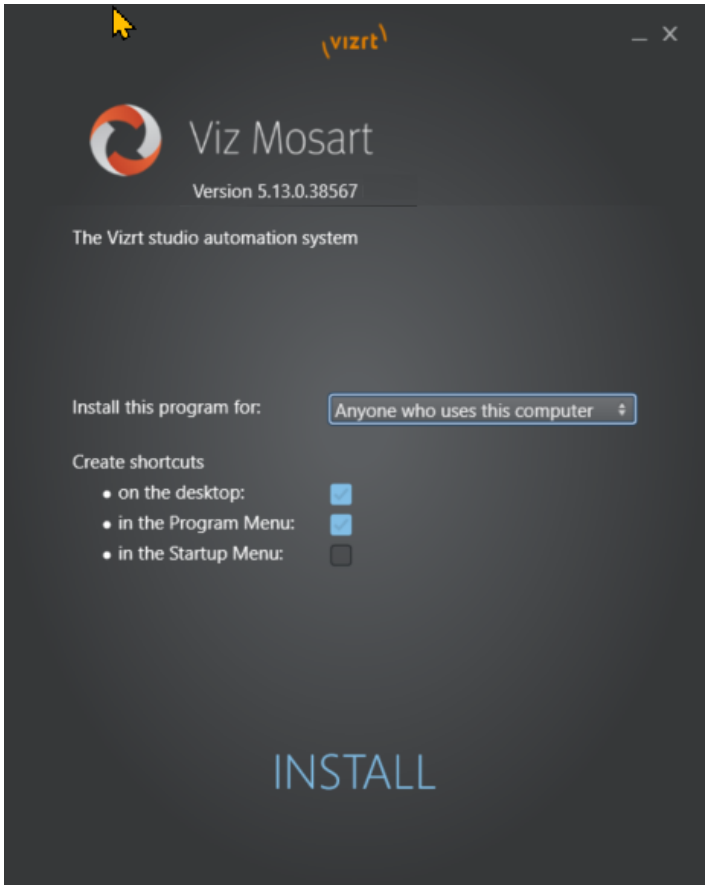
- Use the combined Manus Administrator, *Vizrt.Mosart.ManusAdministrator.exe*.
- For details of the legacy software, refer to versions of Viz Mosart documentation prior to 5.13.

6.1.1 Installation

To install Manus Administrator

Manus Administrator forms part of the standard Viz Mosart (server) installer.

- See the section Viz Mosart Installer Files for details of a standard installation, that includes Manus Administrator



Tip: Type **help** in the Manus Administrator console window to see which commands you can use. See Manus Admin Commands.

See also Avid iNews Web Service for Status Feedback in the Appendix.

6.2 Configuration Utilities

6.2.1 Centralized Settings Menu

A significant Viz Mosart ease-of-use improvement has been introduced for Manus Administrator.

- All settings, including NRCS connectivity, are now presented in a single **Settings** menu. See section Viz Mosart Settings and Configurations.

To open the Settings menu

From the Manus Administrator console you can open a configuration editor in a new menu:

- Type `settings` in the Manus Administrator console, then ENTER.

For NRCS details see:

- Viz Mosart Settings and Configurations (combined)
    - Settings Editor iNews or
    - Settings Editor MOS
      according to type of NRCS you are using.

6.2.2 Newsroom Settings

- To open the **Newsroom Settings XML Editor**, type `ns` in the console window followed by **ENTER**. For details see:
    - Newsroom Settings Editor.

> **Info:** The Newsroom Settings editor opens in a new window. If you make changes to the configuration, save then restart Manus Administrator for the changes to be applied.

The rest of this section covers these topics:

- Settings Editor iNews
- Settings Editor MOS
- Configuring the MSE Playlist Panel
- Field Mapping
- Manus Admin Commands
- Newsroom Settings Editor
- iNEWS Timer

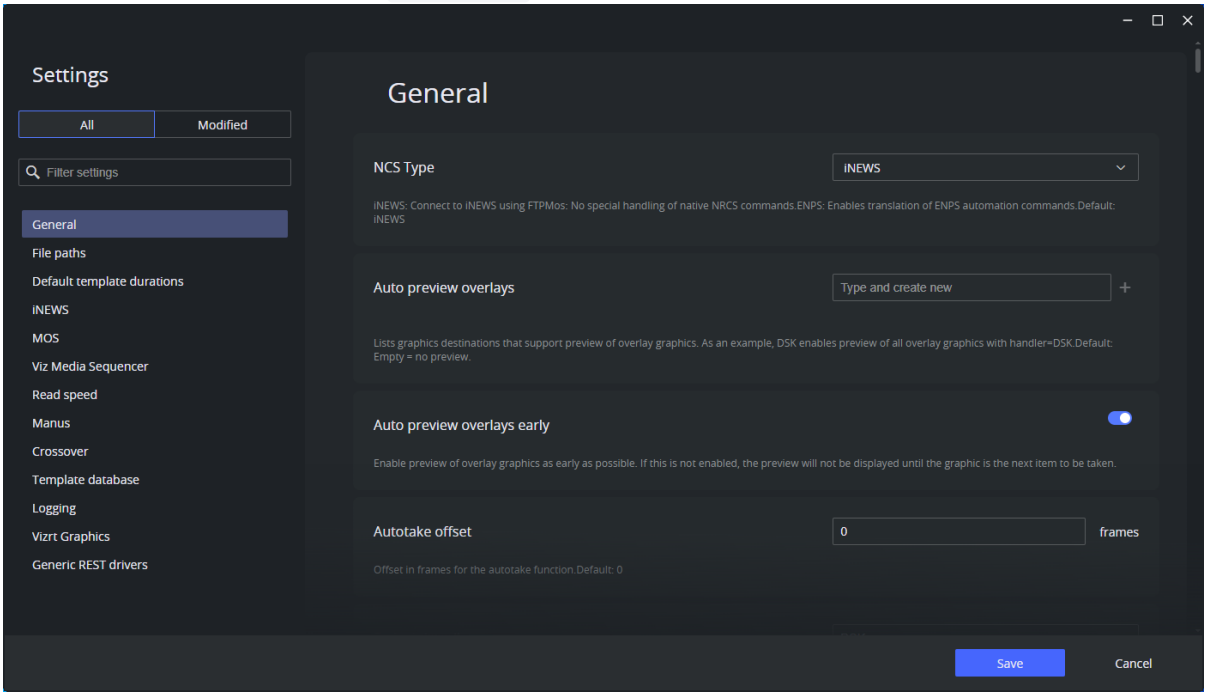
Copyright © 2026 Vizrt

6.3 Settings Editor iNews

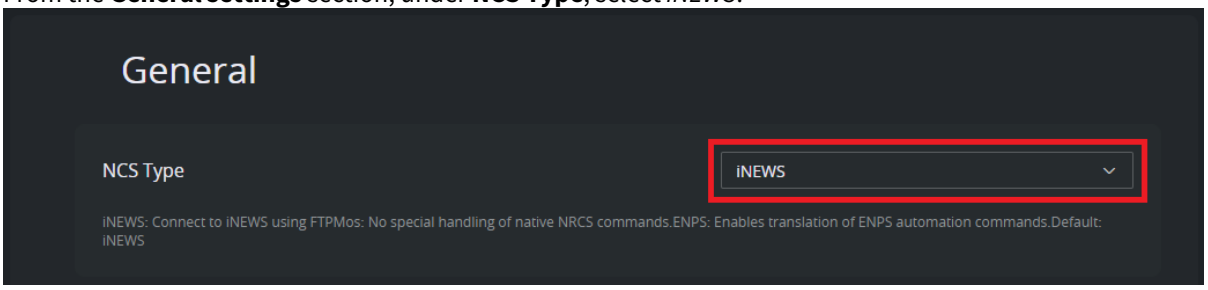
6.3.1 Working with iNEWS Settings

To simplify setting up the connection to iNEWS, the Manus Admin console includes a Settings utility.

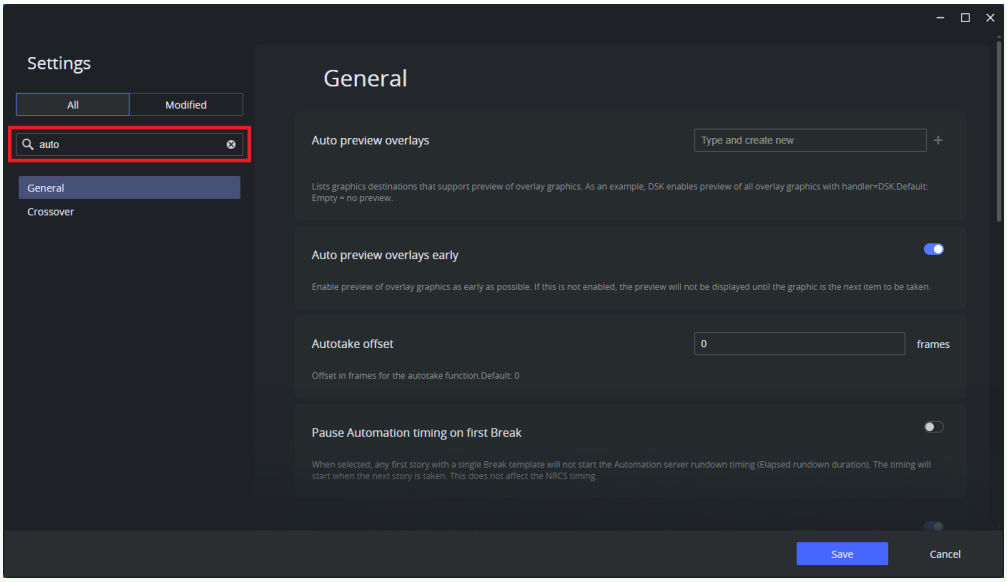
1. Open the Settings utility by typing `settings` followed by **ENTER**.



2. From the **General settings** section, under **NCS Type**, select *iNEWS*.



Tip: You can search for a specific value from the Search box.



The editor comprises several sections, each dedicated to a collection of settings:

- General
- File Paths
- Default Item Durations
- iNEWS
- Viz Media Sequencer
- Read Speed
- Manus
- Crossover
- Template Database
- Logging
- Vizrt Graphics

General

- **NCS Type:** Selecting *iNEWS* presents the configurable values listed here.
- **Auto preview overlays:** List of graphic destinations that support preview of overlay graphics (For example, *DSK* enables preview of all overlay graphics with *handler=DSK).* Default: *Empty* = no preview.
- **Auto preview overlays early**: Enable preview of overlay graphics as early as possible. When *not* enabled, the preview will only be displayed when the graphic is the next item to be taken.
- **Autotake offset:** Sets offset for the autotake function. Default: *0* frames.
- **Default handler name:** For all graphics that have no handler name, Mosart can default to a specific handler for these. You can assign this handler here. Default: *DSK.*
- **Default Lower Third out behaviour:** Default method for taking out a lower third. Default: *TIMECODE*.
- **Force clip editorial time:** Shows Editorial time in GUI. Default: *Off*.
- **Frame rate:** Determines the frame rate of the system. Valid rates *25, 29.97, 50* and *60*.
- **Ignore updates if nothing changes:** Ignores story update from the NRCS if the updated story is assumed to be equal to the current version of the story when set. Default: *Off* (NRCS story updates are *not* ignored).

- **Keep story status on updates from NRCS:** Keeps played-out stories gray if the story is updated in the NRCS. Default: *Not enabled*, do not keep story status.
- **Max number of dynamically inserted items**: The number of story items that will be inserted and stored dynamically on a Mosart story row. Items can, for example, be inserted with a keyboard shortcut. Default: *0* (all items will be stored).

Note: The greater the number of dynamic items that are kept in a story, the slower the system will become. You will need to verify what performance is acceptable for you, depending on your story properties.
If you experience slowness in the Viz Mosart UI while dynamically taking items into preview or program on a running story, you can try initially changing this value to *5* and then verifying on-air performance.
Default value for this setting is *0*, which means that behavior of your system will be unchanged, keeping all manual items.

- For *adlibs* (floats) this solution introduces a potentially undesirable side effect.
  If the adlib is removed from the list of manual items, Viz Mosart will lose the paused position, so that when the adlib is taken again, it will start playing from the *beginning* instead of from the paused position.

- **Metadata creator priority:** Overrides priority timing information from the video file's source. Select between:
  - *Newsroom editorial time*
  - *MAM duration*
  - *Full video clip length*.

Separate IDs with commas. Use the name of the Media Administrator entry. Special names are *NCS* (for information from the NRCS) and *GLOBAL* (global values from media object).
If *Empty*, Viz Mosart inserts *NCS* as the first and *GLOBAL* as last entries.
Default: *Empty.*

- **Minimum clip length**: Sets the minimum visible length of an offline clip in a story in the GUI. When Viz Mosart receives clip information the visible length is updated. Default: *5* (ss:ff).
- **My port:** The port used for other Viz Mosart applications to connect to the Manus Administrator. Default: *8085*.
- **Network exclude:** Manus Administrator automatically enters idle mode on network failure.
  This property is a CSV list of network names to ignore when monitoring networks. Default: *Empty* (monitor all networks).
- **Network include:** Manus Administrator automatically enters idle mode on network failure.
  This property is a CSV list of network names to monitor for network failure. Default: *Empty* (monitor all networks).
- **Offset secondary events with mix delay:** Secondary events follow primary events mix delay.
  For example, a lower third element is delayed with the same mix delay as the package it is to be keyed on.
  Default: *False*
- **Pause Automation timing on first Break:** When enabled, any first story with a single break template will not start the automation server rundown timing (Elapsed rundown duration).
  The timing will start when the next story is taken. This does not affect the NRCS timing.
- **Preload Accessory Cue Delay**: Defines the delay between end of cue the next item and when a pending preload/pretake accessory shall be executed.

Copyright © 2026 Vizrt

Time is in frames. Default = *6* frames, so the accessory preload/pretake functions take place after the cue operation is done in AV Automation, by this number of frames.

- **Reset AutoTake on 'Clear Loop':** Disables the autotake mode automatically when using the clear loop function from the GUI when enabled. Default: *Enabled*.
- **Server description:** User-defined descriptive name of the server that is displayed in the **Timing Display**.
- **Should upgrade:** For *Vizrt Support* use only. Do not change. Default: *False*
- **Startup in idle mode:** Determines whether or not the **Manus Administrator** is idle at start-up. For normal Viz Mosart operation, this setting shall always be *True*, so that the Viz Mosart server starts up in Idle mode. Switching between *Idle* and *Active* mode shall in normal operation, be done from the GUI. Default: *True*
- **Story compare ignore attributes:** Used for debugging purposes only. Default: *Empty*
- **Templates allowing graphic pretake:** Lists template types that allow pretake of overlay graphics elements. Available template types:
    - *ADLIBPIX*
    - *BREAK*
    - *CAMERA*
    - *DVE*
    - *FULLSCREENGRAPHICS*
    - *LIVE*
    - *PACKAGE*
    - *TELEPHONEINTERVIEW*
    - *VOICEOVER*

Default *: PACKAGE* and *VOICEOVER.*
Only the first graphic found on the next item is taken and this is taken immediately regardless of its start time.

> **Info:** For graphic pretake to work, make sure the **Pretake overlay on handler** setting is also configured in the Newsroom Settings Editor.

- **Transitions on Accessories:** Allows transition effects on accessory templates when enabled. If disabled, any transition effects are removed from accessory templates when translating newsroom XML into Mosart XML. This is how it has been, historically.
- **Use TakeOut logic:** Enables the overlay graphic takeout functionality. When two overlay graphic objects with the same graphic IDs are next to each other, the first one is not taken out, only **take in** on the second object is executed.
  When *True*, enables the takeout functionality for overlay graphics. Default: *False.*
- **Advanced Clip Status Display:** When selected, this option activates advanced visualization within the Mosart UI, providing indicators for different clip states. These include, for example,
    - clips present only on mirror servers
    - clips available on the main server but missing from the mirror
    - clips missing on the specified playout server in the template.

When this option is *unselected*, the system defaults to the original visualization method, where a solid blue color signifies clip availability on at least one of the configured video servers in Media Administrator, and a blue and red checkerboard pattern denotes unavailability.
This feature is accessible exclusively in the new combined Manus Administrator settings. Default: *Selected*.

- **Ignore Send Cue Status to NRCS for offline clips:** When selected, the NRCS will not receive CUED and READY status for offline video clips. Default: *Unselected,* do *not ignore*.

- **Refresh media on NRCS update:** When selected, triggers a clip refresh for all clips belonging to a story that is being updated. This usually happens through the NRCS story update mechanism. Ensures that all clip information is synchronized. Default: *Do not trigger refresh*.
- **Time in ms to delay initializing devices while waiting for NRCS to add story:** Set to 1200 to avoid traffic overload (blinking whilst loading). Set to 500 to avoid the first story being set till the end of the rundown. Default: *500 ms*.
- **Use 'Mark template as empty' feature:** This feature allows you to mark empty templates as intentionally empty. Empty templates are treated by Mosart differently in a hierarchical template setup depending if this setting is enabled or not. When a template is marked as *intentionally empty*, it will be used by Mosart as such in the execution and visualization of a template, canceling any corresponding base template implementation. If a template is not marked as intentionally empty, Mosart will search the parent template set chain to find the next available non-empty template for execution and visualization. If this feature is disabled, the same behavior applies as when the template is marked as intentionally empty. Mosart will use the empty template as is, which will cancel any corresponding base template implementation. Default: *Not enabled.* See more details in section Empty Template.

File Paths

- **AV config:** The path to the XML-file containing the audio and video mappings defined in the AV Automation application. Default: `c:\channeltemplates\avconfig.xml`.
- **Channel templates:** The path to the file containing the Viz Mosart templates built in the AV Automation Template Editor. Default: `c:\channeltemplates\channeltemplates.xml`.
- **Manus directory:** The path to the folder containing copies of the internal Viz Mosart rundown. Default: `c:\manus`.
- **Newsroom settings:** The path to the XML-file containing the mappings from newsroom system commands to Viz Mosart templates. Default: `c:\channeltemplates\newsroomsettings.xml`.

Default Item Durations

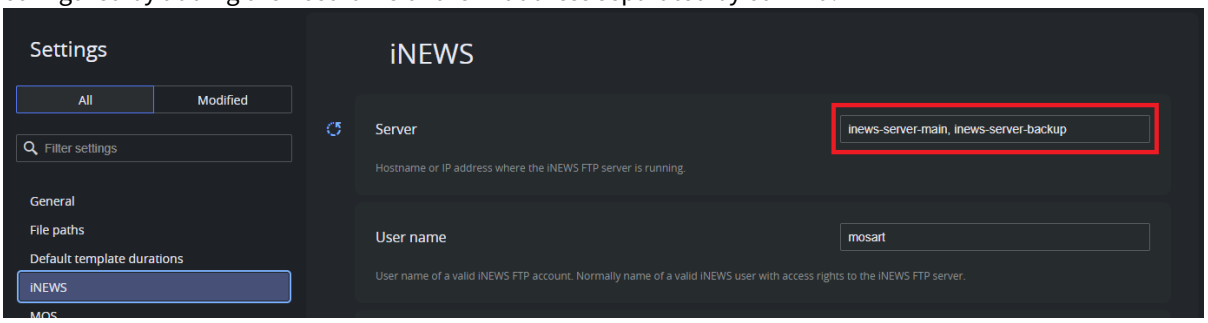
- **Break:** The minimum length given to the **BREAK** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *100* (frames).
- **Item:** The minimum length given to any template type in the rundown. If no time is given in the NCS, then this time is given to the element.
  Default: *100* (frames).
- **Camera:** The minimum length given to the **CAMERA** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **DVE:** The minimum length given to the **DVE** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **Fullscreen graphic:** The minimum length given to the **FULLSCREEN GRAPHIC** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).

Copyright © 2026 Vizrt

- **Live:** The minimum length given to the **LIVE** template type in the rundown. If no time is given in the NCS, then this time is given to the element.
  Default: *125* (frames).
- **Lower third:** The minimum length given to the Lower-third template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *100* (frames).
- **Telephone interview:** The minimum length given to the **PHONO** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **Video clip:** The minimum length given to the **CLIP** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).

iNEWS

- **Server:** Hostname or IP address where the iNews FTP server is running. A backup iNews server can be configured by adding the hostname or the IP address separated by comma:



- **User name:** Username of a valid iNews FTP account. Normally the name of a valid iNews user with access rights to the iNews FTP server.
- **Password:** Password of a valid iNews FTP account. Normally password to a valid iNews user with access rights to the iNews FTP server.
- **FTP Port:** iNews FTP port.
- **Use FTP active mode:**
  Selected: Use FTP Active mode, where Manus Administrator establishes the command channel but the iNEWS server is responsible for establishing the data channel.
  Unselected: Use FTP Passive mode, where Manus Administrator establishes both channels.

> **Info**: In *active* mode, the client starts listening for incoming data connections from the server on port M. It sends the FTP command PORT M to inform the server on which port it is listening. The server then initiates a data channel to the client from its port 20, the FTP server data port.

- *False*: Passive mode where Manus Administrator establishes both channels.

Info: Where the client is behind a firewall and unable to accept incoming TCP connections, *passive* mode may be used. The client then uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server, which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received.

- **FTP active port range:** Ports used for Active mode. The format is *'<low_port>..<high_port>'*, random ports are used when left empty or the format is wrong. Useful when the current machine has certain ports that are blocked or used for other purposes.

Info: Only used when active mode is selected. Similar to the iNEWS environment variable RXDATAPORTRANGE. Because a new connection is opened for each data transfer (listing directories, downloading stories) the port range must be sufficiently wide (~100 wide range works in most cases).

- When the port range is left empty or the format is invalid, random ports are used.
- When the port range is not wide enough, the rundown won't fully load.
- These situations are indicated in the Manus Admin console and in the logs. For example:

```
09:49:00 21 retry download of SHOW.BALAZS.CLIPS 0C5D973C:0001D193:61A0DCC7 story #3
09:49:01 21 error downloading No valid active data port available!
09:49:01 21 retry download of SHOW.BALAZS.CLIPS 145D9745:0001D409:61A10812 random
09:49:01 21 error downloading No valid active data port available!
09:49:01 21 retry download of SHOW.BALAZS.CLIPS 035D9754:0001D5C9:61DD9661 clip
09:49:01 21 error downloading No valid active data port available!
```

- **Prefer FTPS connection:**
  True: FTPS encryption is used if supported by the iNews server. Otherwise it falls back to plaintext FTP communication.
  False: Using plain text FTP communication.

Info: Because FTP uses a dynamic secondary port (for data channels), many firewalls were designed to snoop FTP protocol control messages in order to determine which secondary data connections they need to allow. However, if the FTP control connection is encrypted using TLS/SSL, the firewall cannot determine the TCP port number of a data connection negotiated between the client and FTP server. Therefore, in many firewalled networks, an FTPS deployment will fail when an unencrypted FTP deployment will work. This problem can be solved with the use of a *limited range of ports* for data and configuring the firewall to *open* these ports.
FTP connection type and status are reported in Manus Admin:

- With FTP:

```
12:07:47 15 SITE FORMAT=2nsml.
12:07:47 15 Inews ftpserver (bgo-mos-inews using FTP) connected:
12:07:47 15 NCSStatus set iNews bgo-mos-inews True
```

- With FTPS:

```
12:06:16 14 SITE FORMAT=2nsml.
12:06:18 14 Inews ftpserver (inews7x-a using FTPS) connected:
12:06:18 14 NCSStatus set iNews inews7x-a True
```

- **Working directory:** Initial directory in iNews. Viz Mosart provides access to all rundowns stored hierarchically within this directory.
  Default: *SHOW*
- **Encoding:** Character encoding of the iNews story items. Should match the encoding used in iNews.
  Default: Encoding for the operating system's current ANSI code page.
  Other values: *UTF7, UTF8, UTF32, UNICODE, BIGENDIANUNICODE, ASCII*.
- **Web service connection:**
  Use the provided connection string in the format 'param1=value1;param2=value2;...' to establish a connection with Avid iNEWS Web Services.
  If you prefer not to connect to any web service, leave this field blank.
  - **Mandatory parameters**:
    WebServiceServer=host:port, where you can provide only the host if using the default port (8080).
  - **Optionally parameters**: You can include 'iNewsServer,' 'iNewsUsername,' 'iNewsPassword,' and 'Type,' which are sourced from your iNEWS configurations, if available.
    For 'Type,' valid values are 'iNEWS' and 'MosiNEWS.'
  - **Additional configurations**: These can be made in the AvidInewsWebServiceConfig.xml file, located in the Mosart installation folder under Mosart Server/ConfigurationFiles.
    Any settings configured here will override values found in the configuration file.

> **Info**: For complete setup and configuration of iNews Web Service, please refer to section Avid iNews Web Service for Status Feedback.

Viz Media Sequencer

NCS server: When Showmaker 1.0 is used as NRCS for Viz Mosart, this is where you configure the host address to the Media Sequencer (MSE) that Showmaker is connected to.

MSE Playlist Panel Host: The host address of the Media Sequencer (MSE) used by the **MSE Playlist Panel** in the Viz Mosart UI
Configure the host address of existing MSE host to activate the **MSE Playlist Panel**. The **Playlist Panel** offers users active (touch or click-sensitive) thumbnails of graphics held on an MSE.
The Viz Mosart UI needs this MSE connection defined in order for the user to see the **Playlist Panel** (**View > Floating windows > Playlist Panel)**. See the *Viz Mosart User Guide*, section *Viz Mosart user Interface > MSE Playlist Panel*.

Read Speed

- **Read speed:** Number of words read per minute by the anchor. Applied to the prompter text to determine the duration of an announcement, which is then the length of the green camera bar and the green part of the voice-over bar in the **Viz Mosar**t GUI.
  Default: *145* (words per minute)
  Words are usually taken to be the parts of text separated by space characters. See the setting Use character for read speed word below.
- **Use character for read speed word:** If *True*, each *character* counts as a word for the purpose of calculating the spoken duration of prompter text, see Read speed above.

This option can be useful for languages like Chinese and Thai, where words are not separated by using the space character. If *False* (default), words are separated by space characters.

Manus

- **Default Manus:** Selects the default rundown to use. If set to *True,* the system loads this rundown on startup.
- **Manus expiration time:** Number of days to keep Manus Administrator files. Older files (than that number of days) are deleted when Manus Administrator is started.
  Default: *60* (files are deleted after 60 days).
  Valid alternatives: *-1* (never deleted), *1 and above* (number of days until deleted from `manus` folder).
- **Manus keep file pattern:** If Manus expiration time is given, this property allows a list of file search patterns for files to keep in the Manus Administrator directory.
  Files with names satisfying any of these patterns will be kept even if they are 'too old' according to the **Manus expiration time**
  For example, "*TEST**;DEMO??.xml" keeps all files starting with "*TEST*" and all *DEMO??.xml* files where '?' denotes a wild character.
  Default: *Empty* (no files to keep).

> **Info**: ';' is used to separate the file search patterns.

- **Test manus:** Rundown to be used for maintenance purposes.
- **Use the default selection of Manus:** Initializes the rundown given in the Default Manus setting automatically when enabled. Default: *True*.

Crossover

- **Connection string:** The connection string of the crossover, for example: **controller**=*IP address*, **client**=*crossover*. Default: ""
- **Crossover autotake on switch offset:** Offset in milliseconds to pre-take an autotake when the next story item is after a crossover switch.
  A positive value triggers the autotake before the given duration of the story item.
  A negative value triggers the autotake after the given duration of the story item.
- **Crossover SetNext on switch delay:** Delay of sending set next story from the server running the show when taking control. Default: *500* (milliseconds).

Template Database

- **Use template database:** When true, enables the Template DB functionality, so template type aliases can be exported from DB to `newsroomsettings.xml` at startup, and exporting `newsroomsettings.xml` to DB after saving `newsroomsettings.xml`.
  Default: False.

> **Info:** Enter the **Connection string** before enabling **Use template database**.

- **Connection string:** The connection string for the **Template DB**. For example, for MySQL: server=<hostname>; User Id=<user>;Password=<password>; database=mosarttemplatedb.
  Default: *server=localhost;User Id=root;database=mosarttemplatedb.*
- **Default template DB inserter:** The name to be used for the **insertedby** and **updatedby** columns in the Mosart Template Database.
  Default: *inserter.*

Logging

- **Log level**: Sets the detail level of logging to the log file. *N*ormal, *W*arnings, *E*rrors, Info, *D*etailed.
  Default: *0*.
- **Log MOS communication:** Enable logging of the communication to and from the NRCS.
  Default: *False.*
- **In use:** Enables or disables the logging of events from the Manus Administrator to the log file.
  Default: *True*.
- **WTC Level:** Sets the detail level of logging to the console. *Error, Message, External message, Details*.
  Default: *Details*.
- **Trace internally:** Enables or disables internal tracing to console, for debugging only.
  Default: *False*.
- **Use verbose logging:** Enables or disables verbose logging. If **In use** is set, using verbose the details sent to the log.
  Default: *False*.
- **Ignore verbose events filter:** Semicolon based list of events to ignore in the log when using verbose logging.
  Default: *Empty* (log all events)
- **Pass verbose events filter:** Semicolon based list of events to log when using verbose logging.
  Default: *Empty* (log all events)
- **MSMQ Log limit:** Value to identify when the application should dump the log queue to file. Default: *4023*
- **Path for MMLog:** The path where the **Viz Mosart Log** is stored.
  Default: *MosartLog*.

Vizrt Graphics

- **Enable**: Allow use of Vizrt graphics
- **MSE Host**: The hostname and port number (default *8580*) of the MSE used for playout.
- **Playlist name**: The name of the MSE playlist where Vizrt content from a Mosart show will be added and made ready for playout.
- **MSE Profile name**: Do not change this value.

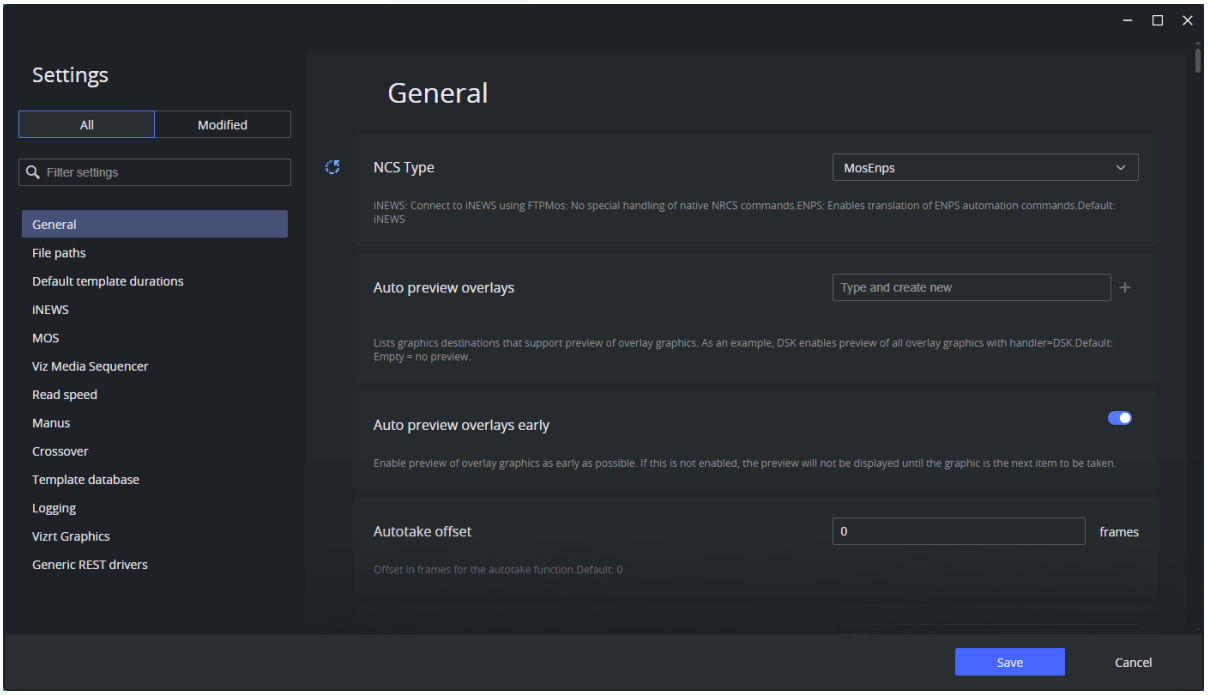
Copyright © 2026 Vizrt

6.4 Settings Editor MOS

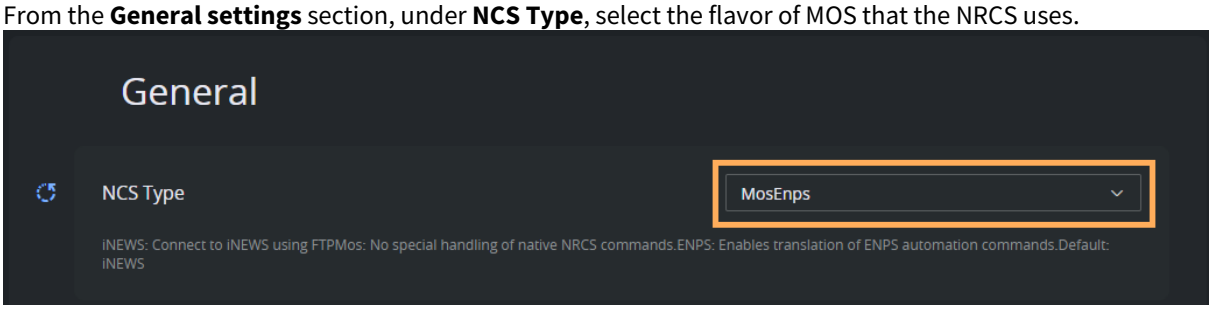
6.4.1 Working with MOS Settings

To simplify setting up the connection to a MOS-based NRCS, the Manus Admin console includes a Settings utility.

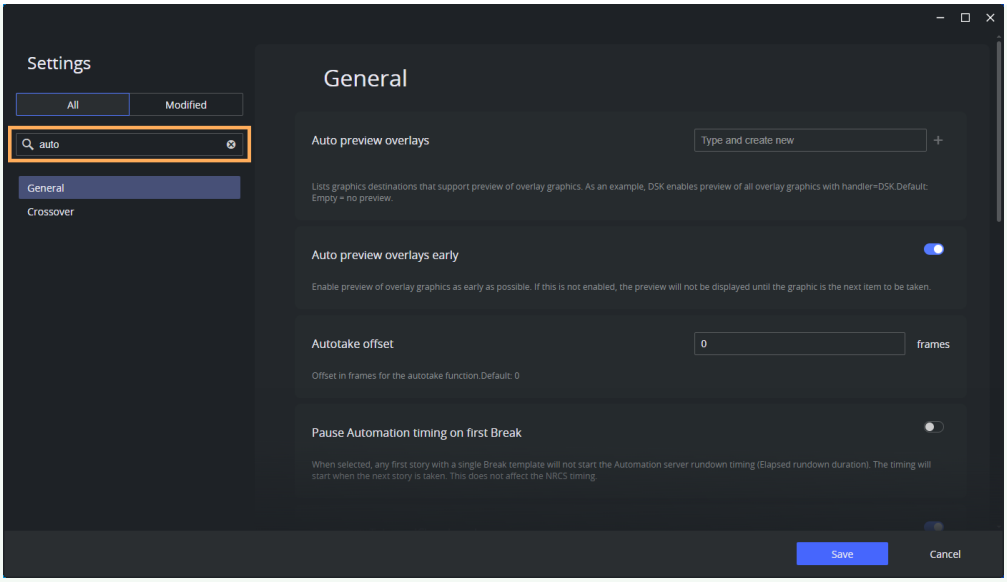
1. Open the Settings utility by typing `settings` followed by **ENTER**.



2. From the **General settings** section, under **NCS Type**, select the flavor of MOS that the NRCS uses.



Tip:You can search for a specific value from the **Search** box.



The editor comprises several sections, each dedicated to a collection of settings:

- General
- File Paths
- Default Item Durations
- MOS
- Read Speed
- Viz Media Sequencer
- Manus
- Crossover
- Template Database
- Logging
- Vizrt Graphics

General

- **NRCS Type:** Generic: No special handling of native NRCS commands. Can be: *Generic*, *DaletPlus*, *ENPS*, *NcPower*, *Octopus*, *Open Media* and *MOSInews*.
  Default: *Generic***.**
- **Auto preview overlays:** Lists graphics destinations that support preview of overlay graphics (i.e. DSK enables preview of all overlay graphics with *handler=DSK).*
  Default: *Empty* = no preview.
- **Auto preview overlays early**: Enable preview of overlay graphics as early as possible. When set to *False*, the preview will be displayed only when the graphics is the next item to be taken.
- **Autotake offset:** Sets offset for the autotake function.
  Default: *0* (milliseconds).
- **Default handler name:** Defines a default handler name for all graphics that have no handler name.
  Default: *DSK.*

- **Default Lower Third out behaviour:**
  Default: *TIMECODE.*
- **Force clip editorial time:** Shows Editorial time in GUI.
  Default: *False.*
- **Frame rate:** Determines the frame rate of the system. Valid rates *25*, *29.97*, *50* and *60*.
  Default: *25.*
- **Ignore updates if nothing changes:** If *True* , a story update from the NRCS is ignored if the updated story is assumed to be equal to the current version of that story.
  Default: *False* (no NRCS story updates are ignored).
- **Keep story status on updates from NRCS:** When enabled, keeps played out stories gray if the story is updated in the NRCS.
  Default: *False.*
- **Max number of dynamically inserted items**: The number of story items that will be inserted and stored dynamically on a Mosart story row. Items can, for example, be inserted with a keyboard shortcut. Default: *0* (all items will be stored).

Note: The greater the number of dynamic items that are kept in a story, the slower the system will become. You will need to verify what performance is acceptable for you, depending on your story properties.
If you experience slowness in the Viz Mosart UI while dynamically taking items into preview or program on a running story, you can try initially changing this value to *5* and then verifying onair performance.
Default value for this setting is 0, which means that behavior of your system will be unchanged, keeping all manual items.

- For *adlibs* (floats) this solution introduces a potentially undesirable side effect.
  If the adlib is removed from the list of manual items, Viz Mosart will lose the paused position, so that when the adlib is taken again, it will start playing from the *beginning* instead of from the paused position.

- **Metadata creator priority:** Select between
  - *Newsroom editorial time*
  - *MAM duration*
  - *Full video clip length*.

Separate IDs with commas. Use the name of the Media Administrator entry. Special names are *NCS* (for information from the NRCS) and *GLOBAL* (global values from media object).
If *Empty*, Viz Mosart inserts *NCS* as the first and *GLOBAL* as last entries.
Default: *Empty.*

- **Minimum clip length:** This value sets the minimum visible length of an offline clip in a story in the GUI. When Viz Mosart receives clip information, the visible length is updated.
  Default: *5* (ss:ff).
- **Network exclude:** Manus Administrator automatically enters idle mode on network failure. This property is a CSV list of network names to ignore when monitoring networks.
  Default: *Empty* (monitor all networks).
- **Network include:** Manus Administrator automatically enters idle mode on network failure. This property is a CSV list of network names to monitor for network failure.
  Default: *Empty* (monitor all networks)

- **Offset secondary events with mix delay:** Secondary events follow primary event mix delay. For example, a lower-third element has the same mix delay as the package on which it is to be keyed.
  Default: *False.*
- **Pause Automation timing on first Break:** When enabled, any first story with a single break template will not start the Automation server rundown timing (Elapsed rundown duration). The timing will start when the next story is taken. This does not affect the NRCS timing.
  Default: *False.*
- **Preload Accessory cue delay**: Defines the delay between end of cue the next item and execution of a pending preload/pretake accessory. The duration is displayed in frames.
  Default: *6* frames, i.e. the accessory preload/pretake functions shall take place after the cue operation is done in AV Automation with an additional and configurable delay of **PreloadAccessoryCueDelay** frames.
- **Reset AutoTake on 'Clear Loop':** Enable this to automatically disable the autotake mode when using the clear loop function from the GUI.
  Default: *True.*
- **Server description:** Used for display only, in **Timing Display**.
  Default: *ControlRoom.*
- **Startup in idle mode:** Determines whether or not Manus Administrator is idle at start-up. For standard Viz Mosart operation, this setting shall always be *True*, so that the Viz Mosart server starts up in *Idle* mode. Switching between Idle and Active mode shall in normal operation be done from the GUI.
  Default: *True.*
- **Story compare ignore attributes:** Used for debugging purposes only.
  Default: *Empty.*
- **Templates allowing graphic pretake:** Lists template types that allow pretake of overlay graphics elements. Available template types: *ADLIBPIX, BREAK, CAMERA, DVE, FULLSCREENGRAPHICS, LIVE, PACKAGE, TELEPHONEINTERVIEW, VOICEOVER*. Default: *PACKAGE, VOICEOVER.*
  Only the first graphic found on the next item is taken. This graphic is taken *immediately* regardless of its start time.

> **Info:** To enable graphic pretake, the **Pretake overlay on handler** setting must be configured in the Newsroom Settings Editor.

- **Use item status to NRCS:** Sends online offline CUE/PLAY/STOP status to the NRCS when enabled. Default: *False*.
- **Advanced Clip Status Display:** When selected, this option activates advanced visualization within the Mosart UI, providing indicators for different clip states. These include, for example,
  - clips present only on mirror servers
  - clips available on the main server but missing from the mirror
  - clips missing on the specified playout server in the template.

When this option is *unselected*, the system defaults to the original visualization method, where a solid blue color signifies clip availability on at least one of the configured video servers in Media Administrator, and a blue and red checkerboard pattern denotes unavailability.
This feature is accessible exclusively in the new combined Manus Administrator settings. Default: *Selected*.

- **Ignore Send Cue Status to NRCS for offline clips:** When enabled, the NRCS will not receive CUED and READY status for offline video clips. Default: *Not enabled*.

- **Refresh media on NRCS update:** When selected, triggers a clip refresh for all clips belonging to a story that is being updated. This usually happens through the NRCS story update mechanism. Ensures that all clip information is synchronized. Default: *Do not trigger refresh*.
- **Time in ms to delay initializing devices while waiting for NRCS to add story:** Set to 1200 to avoid traffic overload (blinking whilst loading). Set to 500 to avoid the first story being set till the end of the rundown. Default: *500 ms*.
- **Use 'Mark template as empty' feature:** This feature allows you to mark empty templates as intentionally empty. Empty templates are treated by Mosart differently in a hierarchical template setup depending if this setting is enabled or not. When a template is marked as intentionally empty, it will be used by Mosart as such in the execution and visualization of a template, i.e. it will cancel any corresponding base template implementation. If a template is not marked as intentionally empty, Mosart will search the parent template set chain to find the next available non-empty template for execution and visualization. If this feature is disabled, the same behavior applies as when the template is marked as intentionally empty, i.e. Mosart will use the empty template as is, which will cancel any corresponding base template implementation. Default: *Not enabled.* For more details see Empty templates.

File Paths

- **AV config**: The path to the XML-file containing the audio and video mappings defined in the AV Automation application.
  Default: *c:\channeltemplates\avconfig.xml*.
- **Channel templates:** The path to the file containing the Viz Mosart templates built in the AV Automation Template Editor.
  Default: *c:\channeltemplates\channeltemplates.xml*.
- **Manus directory**: The path to the folder containing copies of the internal Viz Mosart rundown.
  Default: *c:\manus*
- **Newsroom settings:** The path to the XML file containing the mappings from newsroom system commands to Viz Mosart templates.
  Default: *c:\channeltemplates\newsroomsettings.xml.*

Default Item Durations

- **Break:** The minimum length given to the **BREAK** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *100* (frames).
- **Item:** The minimum length given to any template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *100* (frames).
- **Camera:** The minimum length given to the **CAMERA** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **DVE:** The minimum length given to the **DVE** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).

Copyright © 2026 Vizrt

- **Fullscreen graphic:** The minimum length given to the **FULLSCREEN GRAPHIC** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **Live:** The minimum length given to the **LIVE** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **Lower third:** The minimum length given to the Lower-third template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *100* (frames).
- **Telephone interview:** The minimum length given to the **PHONO** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- **Video clip:** The minimum length given to the **CLIP** template type in the rundown. If no time is given in the NRCS, then this time is given to the element.
  Default: *125* (frames).
- m length given to the CLIP template type in the rundown. If no time is given in the NRCS, this time is given to the element.
- Default: *125* (frames).

MOS

These settings are used when sending Mosart templates as MOS objects to the NRCS. MOS objects can then, for example be used to create items in a rundown (as an alternative to the legacy Mosart ActiveX implementation). You can also send a MOS object containing lower-third information.

- **MOS server:** IP address or hostname of the main newsroom system's MOS gateway. See Notes below.
  Default: *NCSSERVER***.**
- **MOS server backup:** IP address or hostname of the backup newsroom system's MOS gateway.
  See Notes below.
  Default: *Empty*.
- **NRCS ID:** Same form as mosID, but this value is the ID for the newsroom system. See Notes below.
  Default: *NCSSERVER***.**
- **NRCS backup ID:** Same form as mosID, but this value is the ID for the backup newsroom system. See Notes below.
  Default: *Empty*.
- **MOS ID:** MOS identification of this instance of the Manus Administrator. Generic value is mosart.<galleryID>.<stationID>.mos.
  Default: *mosart.mos.*
- **Use backup server:** Enables an NRCS backup configuration. See Notes below.
  Default: *False.*
- **Force MOS protocol version:** Protocol version to use for NRCS communication. Default (blank) setting detects the version from the NRCS, otherwise, use a discrete value, for example, *2.8.3*.
  Default: *Empty***.**
- **MOS lower port:** The MOS protocol communicates on three ports: lower, upper and top. MosUpperPort is MosLowerPort + 1 and MosTopPort is MosLowerPort + 2.
  Default: *10540*

- **Mos server timeout:** Timeout information for MOS ports. Default: *0.0.0.0*.
  - **1. NRCS to Mosart lower port:** NRCS heartbeat sent to Viz Mosart. If the time between received heartbeats exceeds this value, the connection is displayed as *lost*. Value=*0* means no timeout. Default: *0* (seconds).
  - **2. NRCS to Mosart upper port**: NRCS heartbeat sent to Viz Mosart. If the time between received heartbeats exceeds this value, the connection is displayed as *lost*. Value=*0* means no timeout. Default: *0* (seconds).
  - **3. NRCS to Mosart high port:** NRCS heartbeat sent to Viz Mosart. If the time between received heartbeats exceeds this value, the connection is displayed as *lost*. Value=*0* means no timeout. Default: *0* (seconds).
  - **4. Mosart to NRCS:** Timeout value for an established connection between Viz Mosart and the NRCS. If the time from heartbeat is sent to response is received exceeds this value, the connection is displayed as *lost*. Value=*0* means no timeout.
    Default: *0* (seconds).
- **MOS server time zone:** Used when the time zone is not indicated by the NCS Server. Valid values are *Z* (meaning UTC) and *+/-hh[:mm]*, where *hh*and *mm* are double digit hours and optional minutes.
  Default: *Z* (UTC) unless another value is specified.
- **MOS schema:** A string used to identify Viz Mosart items in the rundown. Default: http://www.mosartmedialab.no/schema/mositem.dtd.
- **Default ready to air:** If the value is true, all MOS active rundowns are assumed to be ready to air.
  Default: *False***.**
- **Templates to MOS objects** (see also Mosart Templates Feedback to the NRCS):
  - **Enabled:** Enables sending templates to the NRCS through MOS communication.
    Default: *Enabled*.
  - **GenerateUniqueObjId:** Create a unique identifying name when creating a new MOS Object. The NRCS uses this information to search for and reference the MOS Object.
  - **GroupedByType:** Enable to send all template types as one MOS object with the variants embedded in the objects. Disable to send all template variants as separate MOS objects.
  - **MergedClips:** Determines whether **PACKAGE** and **VOICEOVER** templates are merged as **CLIP** templates.
  - **SendAllTemplateSets:** Sends templates from all template sets to the NRCS when enabled. Disable to send only the default template set.
  - **TemplateChangeWanted:** Set to **true**, the NRCS will receive an update whenever a template is changed or deleted.
- **Story Scope:** Displays either:
  - *Single* = A single story uses one row in the Viz Mosart GUI.
  - *Grouped* = A single story may span over multiple rows in the Viz Mosart GUI.
    Currently only supported for ENPS stories in *Grouped* mode that have the same story name but different story segments; these are then treated as the same story in Viz Mosart.
    Default: *Empty* (Single).
- **MOS reply with Connection MOS ID:** If set, any MOS message sent to the NRCS uses the connection MosId instead of the current mos object ID.
  Default: *False***.**

- **Min. time between story update:** If no update is received from the NRCS within the minimum time stipulated, Viz Mosart forces a story update.
  Default: *1000* (milliseconds).
- **Item ID generator:** Default: *Empty*.
  Mosart generates internal IDs using a combination of rundown, story and item IDs. Some newsroom systems reuse item IDs between story updates. If a new object is inserted before an old object, Viz Mosart will therefore generate the same ID for the new object that the old object had, and the old object will be given a new ID.
  A custom ID generator can be assigned in **Item ID generator**.

Value	Description
(Empty)	This is the default value and uses the default ID generator
ItemAndObjectID	This ID generator appends the object ID to the default ID generator. This setting should be used for Dalet Newswire.

- **Ignore initial synchronization:** Ignores all roReq on startup when synchronizing with the NRCS.
  Default: *False.*
- **Ignore Send Item Status filter:** Lists regular expressions used to prevent sending MOS status back to the NRCS.
  The regular expressions are matched against MosIds of MOS objects. Set to '*' to disable the sending status for all MOS objects.
  Default: *Empty* (send status for all MOS objects).
- **Newsroom tag keep keywords**: A comma (or semicolon) separated list of CAPITALIZED words.
  For example:
  KEEP
  KEEP,INHERIT
  If a newsroom tag value is equal to any of these words, the newsroom tag will be assigned the value of the last Newsroom tag with the same name in the same story.
  Default: *KEEP*.
  See **Keep Newsroom tag (with the same name) within a story** in *section* Newsroom Tags . If "," (only a comma) is used all newsroomtags will be kept.
- **Item status unavailable values:** Lists status values used to identify NRCS content as invalid. Content as graphics or clip objects.
  Default: *NOT_READY, ERROR.*
- **Video server goto delay on retake**: The delay, measured in milliseconds, determining the time taken to position the video server timeline after cueing a pre-roll template for previewing recorded material while Viz Mosart is in Story Recorder mode.
  The configuration of this delay takes into account several factors, including clip length, video server response, and network connection, all of which can influence the cueing time.
  For further details, see section Story Recorder.

Read Speed

- **Read speed:** Number of words read per minute by the anchor. By default, words are defined as the parts of text separated by space characters. See **Use character for read speed word** below. This is applied to prompter text to determine the duration of an announcement and thus the length of the green camera bar and the green part of the voice-over bar in the Viz Mosart GUI.
  Default: *145* (words per minute).

> **Info:** This is a *fallback read rate*, only used when the NRCS System fails to provide a **Read speed** of its own. This function can be tested by changing any **Read speed**="xx" to **Read speed**="" on an item in a local Manus file.

- **Use character for read speed word:** If set to *True*, each *character* counts as a word when calculating the spoken duration of prompter text, see Read speed above. This can be useful in languages like Chinese and Thai, where the space character is not used to separate words. If *False* (default), words are separated by space characters.
  Default: *False.*

Viz Media Sequencer

- **NCS server:** When Showmaker 1.0 is used as NRCS for Viz Mosart, this is where you configure the host address to the Media Sequencer Engine (MSE) that Showmaker is connected to.
- **MSE Playlist Panel Host:** The host address of the Media Sequencer Engine (MSE) used by the **MSE Playlist Panel** in the Viz Mosart UI
  Configure the host address of existing MSE host to activate the **MSE Playlist Panel**. The **Playlist Panel** offers users active (touch or click-sensitive) thumbnails of graphics held on an MSE.
  The Viz Mosart UI needs this MSE connection defined in order for the user to see the **Playlist Panel** (**View > Floating windows > Playlist Panel)**. See *Viz Mosart User Guide*, section *Viz Mosart user Interface > MSE Playlist Panel*.

Manus

- **Default Manus:** This value selects the default rundown to use. The **READYTOAIR** value selects and uses all rundowns indicated by the newsroom system as ready to air.
  Default: **READYTOAIR**
- **Manus expiration time:** Number of days to keep Manus Administrator files. Older files (than that number of days) are deleted when Manus Administrator is started.
  Default: 60 (files are deleted after 60 days).
  Valid alternatives: *-1* (never deleted), *1 and above* (number of days until deleted from the manus folder).
- **Manus keep file pattern:** If Manus expiration time is given, this property allows a list of file search patterns for files to keep in the Manus Administrator directory.
  Files with names satisfying any of these patterns will be kept even if they are 'too old' according to the **Manus expiration time**

For example, "*TEST*;DEMO??.xml" keeps all files starting with "*TEST*" and all *DEMO??.xml* files where '?' denotes a wild character.
Default: *Empty* (no files to keep).

- **Test manus:** Rundown to be used for maintenance purposes.
- **Use the default selection of Manus:** Initializes the rundown given in the **Default Manus** setting automatically when enabled.
  Default: *True*

Crossover

- **Connection string:** The connection string of the crossover. For example: *controller=IP address, client=crossover*.
  Default: *Empty*.
- **Crossover autotake on switch offset:** Offset in milliseconds to pre-take an autotake when the next story item follows a crossover switch. A positive value triggers the autotake before the given duration of the story item. A negative value triggers the autotake after the given duration of the story item.
  Default: *0*.
- **Crossover SetNext on switch delay:** Delay of sending set next story from the server running the show when taking control.
  Default: *500* (milliseconds).

Template Database

- **Use template database:** When true, enables the Template DB functionality, so template type aliases can be exported from DB to `newsroomsettings.xml` at startup, and exporting `newsroomsettings.xml` to DB after saving `newsroomsettings.xml`.
  Default: False.

> **Info:** Enter the **Connection string** before enabling **Use template database**.

- **Connection string:** The connection string for the **Template DB**. For example, for MySQL: server=<hostname>; User Id=<user>;Password=<password>; database=mosarttemplatedb.
  Default: *server=localhost;User Id=root;database=mosarttemplatedb.*
- **Default template DB inserter:** The name to be used for the **insertedby** and **updatedby** columns in the Mosart Template Database.
  Default: *inserter.*

Logging

- **Log level**: Sets the detail level of logging to the log file. *N*ormal, *W*arnings, *E*rrors, Info, *D*etailed.
  Default: *0*.
- **Log MOS communication:** Enable logging of the communication to and from the NRCS.
  Default: *False.*
- **In use:** Enables or disables the logging of events from the Manus Administrator to the log file.
  Default: *True*.

- **WTC Level:** Sets the detail level of logging to the console. *Error, Message, External message, Details*.
  Default: *Details*.
- **Trace internally:** Enables or disables internal tracing to console, for debugging only.
  Default: *False*.
- **Use verbose logging:** Enables or disables verbose logging. If **In use** is set, using verbose the details sent to the log.
  Default: *False*.
- **Ignore verbose events filter:** Semicolon based list of events to ignore in the log when using verbose logging.
  Default: *Empty* (log all events)
- **Pass verbose events filter:** Semicolon based list of events to log when using verbose logging.
  Default: *Empty* (log all events)
- **MSMQ Log limit:** Value to identify when the application should dump the log queue to file. Default: *4023*
- **Path for MMLog:** The path where the **Viz Mosart Log** is stored.
  Default: *MosartLog*.

Vizrt Graphics

The Vizrt graphics setup and workflow in Viz Mosart is simplified with the introduction of a driver that communicates with Vizrt's Media Sequencer (MSE).

> **Info**: This new driver currently supports Augmented Reality (AR) graphics (Viz Arc elements) only.

To manage AR graphics from Viz Mosart

- Refer to the AR graphics section of this publication for details on inserting and controlling AR graphics into a Viz Mosart show.

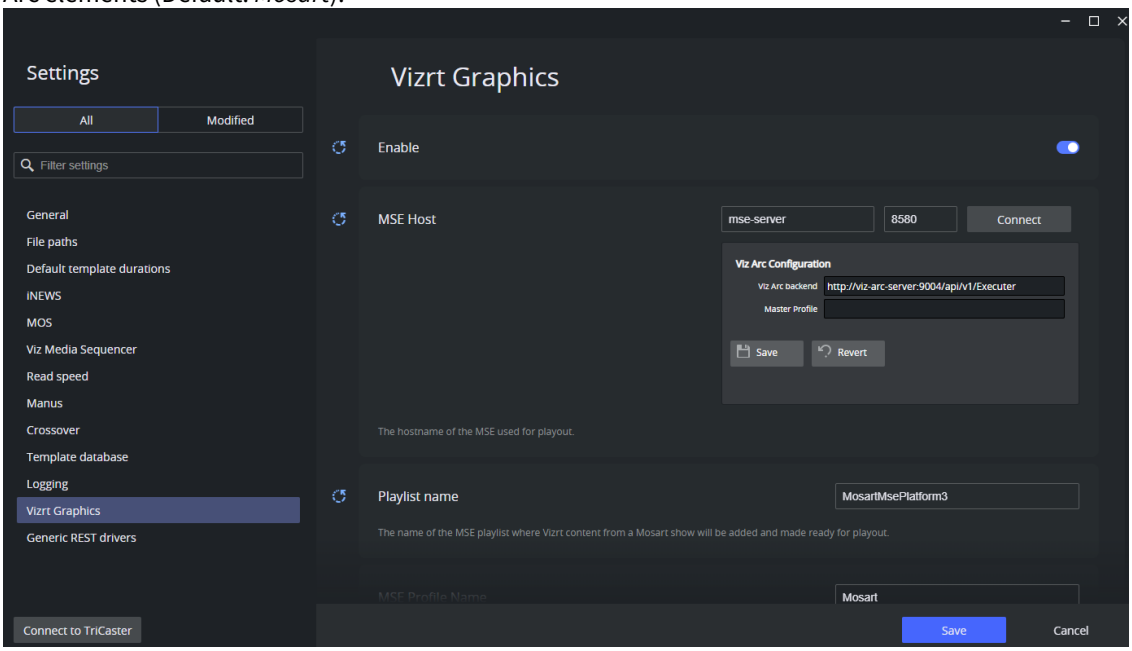
MSE Connection

In the Manus Administrator settings, the new driver is configuring from the **Vizrt Graphics** settings page.

To enable the connection to MSE

1. Open the Settings editor.
2. Navigate to Vizrt Graphics.
3. Select using the following:
   a. **Enable**: Allows the use of Viz Arc elements in the show workflow.
   b. **MSE Host**: The hostname and port number of the MSE used for playout (Default: *8580*).
      i. Click **Connect** to verify the connection and retrieve the Viz Arc settings from MSE.
      ii. When the connection is successful, a form will appear where you can enter the Viz Arc configuration.
      iii. Click **Save** to push the changes to MSE.
   c. **Viz Arc backend**: The URL to the Viz Arc REST API service. Default: `http://mse-server:9004/api/v1/Executer`.
      i. Replace *mse-server* with the machine name where Viz Arc service is installed.
   d. **Master Profile**: The profile configured in Viz Arc.

e. **Playlist name**: The name of the MSE playlist where Vizrt content from a Mosart show will be added and prepared for playout (Default: *MosartMsePlatform*).
f. **MSE Profile name**: The profile in MSE used for playout of elements. This setting is not relevant for Viz Arc elements (Default: *Mosart*).



6.4.2 Notes

- If there is more than one main newsroom system, the setting parameters **MOS server**, **MOS server backup**, **NRCS ID**, and **NRCS backup ID** should contain space-separated lists of values.
  These values must be in the same order.
- If some of the newsroom systems do not have a backup system put them last in the list, leaving the corresponding values for **MOS server backup** and **NRCS backup ID** *blank*.
- If two newsroom systems are backup for each other, both must be listed as both *main* and *backup*. If one system is backup for more than one *main* system, it must be listed as *backup* for each system.
- If different newsroom systems need to give Viz Mosart different MOS IDs, the settings parameter **MOS ID** should also be a space-separated list of values, in the same order as for **NRCS ID**.
- Use only a single space between list values. All values are case sensitive.
- Story Recorder status feedback to the NRCS is supported.

6.5 Configuring the MSE Playlist Panel

The **MSE Playlist** panel is presented on the Viz Mosart UI, offering users active (touch or click-sensitive) thumbnails of graphics held on an MSE (Media Sequencer).

> **Note**: For end user actions, please refer to the *Viz Mosart User Guide*, section **Viz Mosart User Interface > MSE Playlist Panel**.

- User Access to the MSE Playlist Panel
- Host Address of the Media Sequencer in the Viz Mosart UI

6.5.1 Managing the MSE Playlist Panel

User Access to the MSE Playlist Panel

To configure MSE Playlist Panel access from Manus Administrator

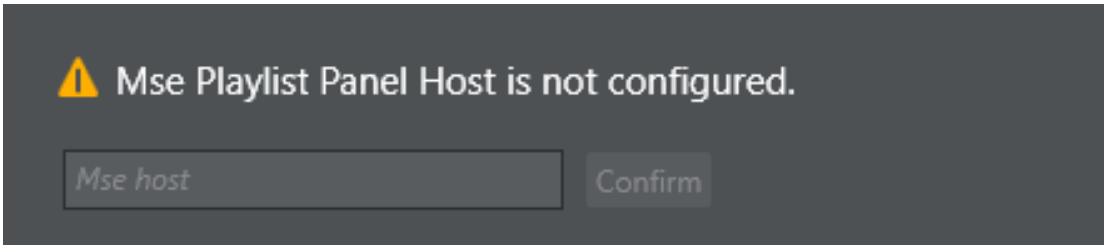
The Viz Mosart UI needs a connection to an active MSE. To activate the **MSE Playlist** panel, you must configure which MSE to use.

1. Open the Viz Mosart Manus Administrator (console) window.
2. Type `settings`
3. Scroll down to **Viz Media Sequencer**.
4. Modify the value of **Mse Playlist panel host** (use either hostname or IP address).
5. Click **Save**.
6. Restart the Manus console.

Host Address of the Media Sequencer in the Viz Mosart UI

To configure MSE access using the Viz Mosart client

- If the **MSE Playlist** panel is visible in the Viz Mosart UI, you can add the host address of the Media Sequencer directly.
  Either host name or IP address can be used.



6.6 Field Mapping

The custom MOS object mapping is a setup which extracts field values from foreign MOS objects and translates these into Viz Mosart fields. This is done in the `newsroomsettings.xml` configuration file. This simplifies integration, as no new code needs to be deployed.

- The Tool Set
- What to Configure
- Target Fields
- Mapping Source Fields to Target Fields
- Examples
  - Video Server Item
  - Graphics Item
  - Harris Video Objects in Octopus
  - Regex: Extracting and Transforming Values

6.6.1 The Tool Set

- Part of the file `newsroomsettings.xml` contains default configurations.
- Select content from the XML using simple XPath 1.0 expressions.
- Concatenate the XPath result using a separator.
- Replace parts of the result using static or dynamic variables.
- Apply conversion rules for the specific field type.

6.6.2 What to Configure

- Reuse or modify existing MOS field mapping in `newsroomsettings.xml`.
- Add MOS Id map entry using the Newsroom Settings Editor.
  - **Newsroom alias**: MOS ID of foreign item.
  - **value**: Field mapping entry name.
  - **appendContent**: Appends the complete MOS object to the internal Viz Mosart <item /> as a _<content /> _sub child.
  - **embeddedMosartItem**: Signal if the item has a Viz Mosart item embedded.

6.6.3 Target Fields

Target fields is a list of the fields used when controlling a video server or graphics elements.

Video Server

- **clip_hirespath:** The name or ID of the video file.
- **clip_description:** A human-readable description of the video file (optional).
- **clip_dur:** The duration of the video file.
- **clip_mark_in:** The start frame of the video file (optional).

Graphics Systems

- **graphics_id:** The ID of the graphics item.
- **graphics_description:** A human-readable description of the graphics item.
- **tc_in:** Planned in-time for the item, relative to the parent item.
- **tc_dur:** Planned duration for the item.
- **graphics_profile:** The graphics concept context that this item should be forced to play in.
  This value is forwarded to graphics systems supporting concepts for graphics when cueing and playing an item.
- **graphics_category:** Used in overlay graphics Take Out Logic to identify graphics that should be kept at story item transitions. If not set, **graphics_id** is used for identification.
- **channel:** The channel to play the item, i.e. **DSK**, **WALL** etc.
- **take: MANUAL** or **AUTO**. If **MANUAL** then **tc_in** is **0** and only **tc_dur** is used.
- **outBehaviour:**
  - **TIMECODE** = the element is taken out relative to the current primary element using **tc_dur**.
  - **BACKGROUNDEND** = the element is taken out when switching primary story element.
  - **STORYEND** = the element is taken out when switching from one story to the next.
  - **OPENEND** = the element is taken out when reloading the rundown or by manually taking the element out.
- **auto_continue:** true/false (only for Orad, to send PlayWithoutPause command).

6.6.4 Mapping Source Fields to Target Fields

The following mapping `<mosmap id=”VENDOR” elementtype=”CLIP”> </mosmap>` contains the attribute descriptions:

- **Id:** Value used in the MOS map section of the newsroomsettings.xml.
- **Elementtype:** CLIP for video servers, GRAPHICS for graphics systems.

And `<fieldmapping />` contains the attribute descriptions:

- **fieldname:** Viz Mosart target field name.
- **mospath:** A simple XPath 1.0 expression to the location of the source value.
- **separator:** Value used for joining multiple returned values for the mospath.
- **fieldtype:**
  - **FIELDS** – the source value is in video fields.
  - **FRAMES** – the source value is in video fields.
  - **TIMECODE** – the source value is in time code format. Use mask attribute to identify format.
  - **STATIC** – the value for the fieldname has no mospaths and should be as defined in the value attribute.
  - **REPLACE**
  - **ADJUST**
- **value:** The value to use if fieldtype is *STATIC*.
- **valuetype:** *TEXT*, *TIMECODE*.
- **mask:** Value on time code format. *hh:mm:ss* or *hh:mm:ss:ff*.
- **mustexist:** Set to *true* if the Viz Mosart field should only be added if a non-empty value exists.
- **regex:** Regular expression applied to the extracted value. Can be used in two modes:
  - (1) with **matchindex** to extract a specific capture group
  - (2) with **replacepattern** to perform regex substitution using capture group references.

- **matchindex:** Selects which capture group to extract from the regex result.
  0 returns the entire match.
  1 returns the first capture group.
  2 the second, etc.
  If omitted (defaults to *-1*), the entire match is returned. Ignored when **replacepattern** is specified.
- **replacepattern:** A replacement string used with **regex** to perform regular expression (regex) substitution.
  Supports capture group back-references (`$1` , `$2` , etc.).
  When specified, instead of extracting a single match, the code uses the **regex** pattern to extract values from the MOS object and apply the regex matches in the **replacepattern** string.
  This is useful for constructing new values by recombining captured parts with static text.
- **overwrite:** Overwrites if a field already exists if set to *true*. Default is *false*.

6.6.5 Examples

Video Server Item

The example below is of a Quantel video item:

Source MOS XML (Quantel MOS item)

```
<mos>
    <itemID>3</itemID>
    <itemSlug>New Row 3 CVD 1-3</itemSlug>
    <objID>15160::559</objID>
    <mosID>QUANTEL</mosID>
    <mosAbstract>Flug über die Alpen fertig2 1:00</mosAbstract>
    <abstract>Flug über die Alpen fertig2 1:00</abstract>
    <objDur>1500</objDur>
    <objTB>25</objTB>
    <objSlug>Flug über die Alpen fertig2</objSlug>
</mos>
```

Info: The objID is constructed from two numbers where the first is the Quantel clip id. The second number is the Quantel zone.

Viz Mosart MOS mapping

```
<mosmap id = "QUANTEL" elementtype = "CLIP" idasindex = "false">
    <fieldmapping fieldname = "clip_hirespath" mospath = "//objID" regex = "^\d+"/>
    <fieldmapping fieldname = "clip_description" mospath = "//objSlug"/>
    <fieldmapping fieldname = "clip_mark_in" fieldtype = "static" value =
"00:00:00:00" valuetype = "TIMECODE"/>
    <fieldmapping fieldname = "clip_dur" fieldtype = "FIELDS" mospath = "//objDur"
valuetype = "TEXT"/>
```

```
</mosmap>
```

Target Viz Mosart XML

```
<fields>
    <field name = "clip_hirespath" value = "15160"/>
    <field name = "clip_description" value = "Flug über die Alpen fertig2 1:00"/>
    <field name = "clip_mark_in" value = "00:00:00:00"/>
    <field name = "clip_dur" value = "750"/>
</fields>
```

Graphics Item

The example below is of an XPression graphics item:

Source MOS XML (XPression MOS item (modified and stripped to fit screen))

```
<mos>
    <itemID>11</itemID>
    <objID>{01E4E214-2414-4C71-956F-6587E3FD1E4B}</objID>
    <mosID>XPRESSION</mosID>
    <mosAbstract>MAIN KEY Name (B2B): This is a test | Test</mosAbstract>
    <itemChannel>1</itemChannel>
    <itemEdStart>0</itemEdStart>
    <itemEdDur>0</itemEdDur>
    <itemTrigger>CHAINED</itemTrigger>
    <macroOut>NONE</macroOut>
    <mosExternalMetadata>
        <mosPayload>
            <gfxtype>CG</gfxtype>
            <itcTimeIn>500</itcTimeIn>
            <itcTimeDur>250</itcTimeDur>
        </mosPayload>
    </mosExternalMetadata>
</mos>
```

Viz Mosart MOS mapping

```
<mosmap id = "XPRESSION" elementtype = "GRAPHICS">
    <fieldmapping fieldname = "graphics_id" mospath = "(//itemID | //objID)"
separator = "-" fieldtype = "REPLACE" value = "{STORYID}-{MOSPATH}"/>
    <fieldmapping fieldname = "graphics_description" mospath = "//mosAbstract"/>
    <fieldmapping fieldname = "tc_in" fieldtype = "FRAMES" mospath = "//itcTimeIn"
valuetype = "TEXT"/>
```

```
    <fieldmapping fieldname = "tc_dur" fieldtype = "FRAMES" mospath = "//
itcTimeDur[text()!='0']" valuetype = "TEXT"/>
    <fieldmapping fieldname = "outBehaviour" mospath = "//macroOut"/>
    <fieldmapping fieldname = "channel" fieldtype = "STATIC" mospath = "//
gfxtype[text()='CG']" value = "DSK" valuetype = "TEXT" mustexist = "true"/>
    <fieldmapping fieldname = "channel" fieldtype = "STATIC" mospath = "//
gfxtype[text()='OTS']" value = "WALL" valuetype = "TEXT" mustexist = "true"/>
    <fieldmapping fieldname = "channel" fieldtype = "STATIC" mospath = "//
gfxtype[text()='FS']" value = "FULL" valuetype = "TEXT" mustexist = "true"/>
    <fieldmapping fieldname = "channel" fieldtype = "STATIC" mospath = "//
gfxtype[text()='OTHER']" value = "FULL" valuetype = "TEXT" mustexist = "true"/>
</mosmap>
```

Target Viz Mosart XML

```
<item>
    <fields>
        <field name = "graphics_id" value =
"VER7ENPS1;P_VANLAB\W\R_F3144D55-78F2-494B-863ED060A74D92F7;D02D7434-58BB-43D6-
B03601E7B7BBC1E7-4-{1ECA795F-101D-4301-A559-2A1392F3AA2D}" fieldtype = "TEXT"/>
        <field name = "graphics_description" value = "BANNER:Chan1: | Manual |
Timed(---):TIMED" fieldtype = "TEXT"/>
        <field name = "tc_in" value = "" fieldtype = "TEXT" inputmask = "ff" default
 = "00:00"/>
        <field name = "tc_dur" value = "125" fieldtype = "TEXT" inputmask = "ff"
default = "00:00"/>
        <field name = "outBehaviour" value = "TIMED" fieldtype = "TEXT"/>
        <field name = "channel" value = "DSK" fieldtype = "TEXT"/>
    </fields>
</item>
```

Harris Video Objects in Octopus

How to configure MOS mapping of a Harris video object in Octopus:

Source MOS XML

```
<storyItem>
    <itemID>2-1</itemID>
    <itemSlug>0406 ZN MUNDE 5 WIN 3 PM NEW</itemSlug>
    <octext_elemLabel>VO</octext_elemLabel>
    <objID>0406 ZN MUNDE 5 WIN 3 PM NEW</objID>
    <mosID>ZEE.HARRIS.MOS</mosID>
    <itemTrigger>MANUAL</itemTrigger>
    <octext_elemType>vo</octext_elemType>
</storyItem>
```

- Update the `newsroomsettings.xml` file by opening it in a text editor and adding the below *mosmap* XML entry in the `<mosmaps />` section (as a child to the `<mosmaps />` element node)

Viz Mosart MOS mapping

```
<mosmap id="HARRIS" elementtype="CLIP" idasindex="false">
        <fieldmapping fieldname="clip_hirespath" mospath="//objID" />
        <fieldmapping fieldname="clip_description" mospath="//itemSlug" />
        <fieldmapping fieldname="clip_mark_in" fieldtype="static" value="00:00:00:00"
 valuetype="TIMECODE" />
<!--
        <fieldmapping fieldname="clip_dur" fieldtype="FIELDS" mospath="//itemEdDur"
valuetype="TEXT" />
-->
</mosmap>
```

- Open the newsroomsettings.xml file in Mosart Server application using the **ns** command.
- Open **Edit -> MOS Id mapping** from the menu and add an entry where the value in Newsroom alias is the MOS ID of the Harris item, (in the above example ZEE.HARRIS.MOS), and the **Value** column is set to **HARRIS** (the id of the mosmap above). If the columns **appendContent** and **embeddedMosartItem** are shown, they should both have the value **false**.

Regex: Extracting and Transforming Values

Example A: Extracting a capture group with `matchindex`

This example extracts just the numeric ID from a backslash-delimited `objID`:

Source MOS XML

```
<storyItem>
    <itemID>test-item-3</itemID>
    <objID>FOLDER\99999</objID>
    <mosID>TESTMOS</mosID>
    <mosAbstract>Test Description</mosAbstract>
    <mosExternalMetadata>
        <mosScope>PLAYLIST</mosScope>
        <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
        <mosPayload>
            <mosarttemplate>
                <type name="LOWERTHIRD">
                    <variant name="TEST"/>
                </type>
            </mosarttemplate>
        </mosPayload>
    </mosExternalMetadata>
```

```
</storyItem>
```

Viz Mosart MOS mapping

```
<mosmap id="TEST" elementtype="GRAPHICS">
    <fieldmapping fieldname="graphics_id" mospath="//objID"
                  regex="^(\w+)\\(\d+)$" matchindex="2" />
    <fieldmapping fieldname="graphics_description" mospath="//mosAbstract" />
</mosmap>
```

The regex `^(\w+)\\(\d+)$` defines two capture groups. `matchindex="2"` selects the second group (the digits after the backslash).

Target Viz Mosart XML

```
<fields>
    <field name="graphics_id" value="99999" />
    <field name="graphics_description" value="Test Description" />
</fields>
```

Example B: Building a path with `replacepattern`

This example extracts parts from an `objID` and constructs a full file path using the `replacepattern` attribute:

Source MOS XML (Chyron/CAMIO item)

```
<storyItem>
    <itemID>ee294520-db64-11f0-94cc-c70081fe9c41:SPEC_NEWS_FL_TB-00010309</itemID>
    <itemSlug> - 00010309(Monitor Plasma Horizontal HMON) GENERIC SINGLE BOTTOM
LINE</itemSlug>
    <objID>SPEC_NEWS_FL_TB\00010309</objID>
    <mosID>CHYRONMOS</mosID>
    <mosAbstract>(Monitor Plasma Horizontal HMON) GENERIC SINGLE BOTTOM LINE</
mosAbstract>
    <mosExternalMetadata>
        <mosScope>PLAYLIST</mosScope>
        <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
        <mosPayload>
            <mosarttemplate>
                <type name="LOWERTHIRD">
                    <variant name="CHYRON"/>
                </type>
            </mosarttemplate>
        </mosPayload>
    </mosExternalMetadata>
</storyItem>
```

Viz Mosart MOS mapping

```
<mosmap id="CHYRON" elementtype="GRAPHICS">
    <fieldmapping fieldname="graphics_id" mospath="//objID"
                  regex="^(.*?)\\(.*)$"
                  replacepattern="I:\CAMIO4\$1\Messages\$2" />
    <fieldmapping fieldname="graphics_description" mospath="//mosAbstract" />
</mosmap>
```

The regex `^(.*?)\\(.*)$` captures `SPEC_NEWS_FL_TB` as `$1` and `00010309` as `$2`. The `replacepattern` recombines them into a full path.

Target Viz Mosart XML

```
<fields>
    <field name="graphics_id" value="I:\CAMIO4\SPEC_NEWS_FL_TB\Messages\00010309" />
    <field name="graphics_description"
           value="(Monitor Plasma Horizontal HMON) GENERIC SINGLE BOTTOM LINE" />
</fields>
```

Note: When `replacepattern` is specified, the `matchindex` attribute is ignored. The two approaches are mutually exclusive -- use `matchindex` to extract a single capture group, or `replacepattern` to construct a new value from multiple groups.

6.7 Manus Admin Commands

These console commands are only intended for advanced users, and principally for troubleshooting. They shall never be used on a live system.

> **Warning:** Use of these commands may adversely affect system performance. Unauthorized use of these commands may invalidate a contracted Vizrt Support agreement.

tmpPrint: Print the manus

pcol: Print col1

setrundown, **SR**, **CD [name]**: Set a new rundown

TESTUPDATE: Start test timer

PRINTRUNDOWNS: Print newsroom rundowns

setManusDirectory: Set the directory in which manus are stored

VERSION Prints the current version

START: Start or stop the timeline
-FILEWATCHER -FW Starts the application in file watcher Modus

R: Release background

INIFILEPATH: Set ini file

INEWSFTPSERVER [servername] [username] [password]: Set the ftp connection to iNEWS FTP server, you must restart

GTS, **G**: Go to next story

NONEAUTOS: List none auto lower thirds

S: Skip next story

WQ: Write queue list

INEWSWORKINGDIRECTORY: Set the working directory

INEWSDEFAULTRUNDOWN: Set the default Showpath

LOG: Collect logs for all connections

SETTINGS: Show the Settings dialog

NEWSROOMSETTINGS, **NS**: Show Settings dialog for newsroomsettings

LOG: Write current log to file

SHOWLOG: Show logging elements

HIDELOG: Hide logging elements

PARSENATIVESTORY, **PNS** - Parse a native NRCS file to Mosart XML

PS [0] [0]: Print story or story item

PST: Print states

READSPEED [0]: Set read speed in words per minute

RELOADMANUS, **RL**, **RELOAD**: Reload current manus

RELOADTEMPLATES, **RT**: Reload channeltemplates

RESYNC: Resync rundown with NRCS

LM, **LOADMANUS [filename**]: Load manus from file

LISTCLIPS: List all items in media cache

TRACE: Activate/deactivate the Trace listener

SAVE [path]: Save the rundown as a file.

STEST Y or **N** millisec

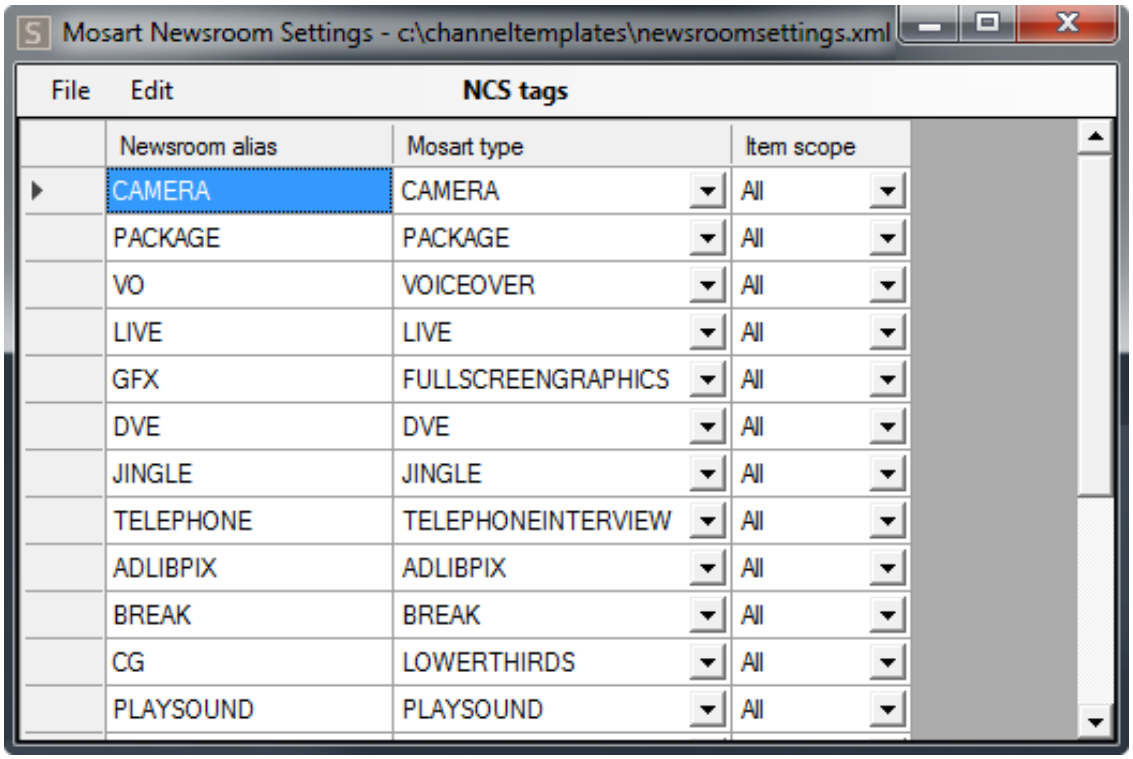
SS, **STUDIOSETUP**: Set the current studio setup

SAVEASSEQUENCE [sequencename as string]: Save the rundown as a sequence

DUMPNCSSTORY, **DNS [storyname] /p[destination filepath]**: Dump a story in native NSML Format

DUMPNCSRUNDOWN DNR /p[destination filepath]: Dump the current rundown in the native NSML Format

6.8 Newsroom Settings Editor



This section contains the following topics:

- Working with the Newsroom settings
  - Configuring NRCS Tags
  - Newsroom Settings Editor - Edit Menu
  - Story External Metadata
  - Graphic Destination Letters
  - Properties
- Viz Mosart Object Structure
  - Vizrt MOS Item With Payload Data Nodes
  - Vizrt MOS Item Without Payload Data Nodes
- Viz Mosart and Viz Content Pilot (VCP) MOS Item Integration
- MOS ID Mapping
- Additional Newsroom Settings Examples
  - Defining Default Values for Lower Thirds
  - iNEWS: Overriding Default Field Names when Parsing Stories
  - iNEWS: Presenter Instructions

Copyright © 2026 Vizrt

6.8.1 Working with the Newsroom settings

Configuring NRCS Tags

To open the Newsroom Settings XML editor

1. Open the **Manus Administrator** console
2. Type `ns` followed by **ENTER**.
3. From the menu bar **Edit > NCS Tags**.

To save changes

Any changes made to the Newsroom Settings configuration must be saved:

1. Either
   a. Go to **File > Save**, or,
   b. Close the **Newsroom Settings Editor** and select **Save**.
2. Restart **Manus Administrator** for changes to apply.

Newsroom Settings Editor - Edit Menu

The **Edit** menu lists the configuration categories that are available.

- **NCS tags:** Menu for assigning the NRCS command aliases from the newsroom system to the corresponding Viz Mosart type. Each type can have multiple aliases. For example, *Camera* type can have aliases **CAM, CAMERA, KAM, KAMERA**, and so on.
- **NCS omit stories:** List of stories that should be ignored, and not displayed in the Viz Mosart GUI.
- **NCS omit tags:** (iNews only) List of iNews grommet commands that are ignored in Viz Mosart.
- **NCS breakline stories:** List of story names that are automatically converted to a Viz Mosart break line story when the story is tagged as *break*.
- **NCS accessory stories:** Viz Mosart interprets a story with this title as a story with accessories only.
- **NCS lowerthird mapping:** List for converting lower third template names from what is written in the NRCS (origin) to the required template (translated) in the graphics system.
  Transtype can be *begins, exact* or *contains.*
- **Timecode character:** Settings for textual analysis of timecode character, start character and split character.
- **Keywords:** Translation of **CUT** *,* **MIX** *,* **WIPE** and **EFFECT** transition keywords.
- **Lower third keep while background:** Lower third variants that are kept on air for the duration of the parent Viz Mosart template.
- **Lower third keep while story:** Lower third variants that are kept on air for the duration of the parent story.
- **Lower third none auto out:** Lower third variants that are kept on air until replaced by another lower third of the same variant. Can be taken out with a keyboard shortcut (for example **HOME**).
- **Lower third keep until manual taken out:** Deprecated.
- **Pretake overlay on handler:** Name of the overlay handlers that automatically take the first lower third element in the next story on item types configured in Manus Administrator Settings, **Template allowing graphic pretake** (same configurations for MOS Settings Editor MOS or iNews*).*

The graphic is taken immediately regardless of its start time.
In the following example, only the graphics with destination **DSK**, **WALL** or **WALL_2** are pretaken.
Example

```
<pretakeoverlayhandlername>
      <handlername name="DSK" />
      <handlername name="WALL" />
      <handlername name="WALL_2" />
</pretakeoverlayhandlername>
```

- **NCS device shortening:** (iNews only) A list of aliases for **clipname** or **clip_hirespath** values that can be used as newsroom tags in iNews grommets.
- **Parenthesis:** Start and end parenthesis. Content within these parentheses is translated to Viz Mosart commands. A value of (*****) translates all commands in the form (**** COMMAND *****) as the ***** (asterisk) value is interpreted as one or more asterisks.
- **Story External Metadata:** The NRCS can send external story values in a container called **mosExternalMetadata** (MEM). These can be translated to Viz Mosart content, as described in section Story External Metadata below.
- **Graphic Destination Letters:** Used to identify the output and behavior of graphic elements. See Graphic Destination Letters below.

Info: Only applies to Vizrt VCP MOS items and the use of the Viz Mosart extended data element description.

- **Lower thirds type translation:** (ENPS only) CG commands. Use this to translate a CG item to a Viz Mosart primary story element
    - **templatetype**: CG template ID to translate
    - **type**: Viz Mosart primary story element
    - **variant**: Viz Mosart variant of the primary story element
    - **transtype**: begins, contains or exact. Describes the usage of the templatetype value when searching the CG item.
- **Newsroom tag to lowerthird channel map:** Translate the handler name of a graphic item into the value defined by this mapping in the newsroomsettings.
  For example, with the following configurations in newsroomsettings.xml:

```
<inewstags>
<tags>
   <tag name="CAMERA" type="CAMERA" scope="All" />
   <tag name="PACKAGE" type="PACKAGE" scope="All" />
   <tag name="VOICEOVER" type="VOICEOVER" scope="All" />
   <tag name="VOICE OVER CLIPS" type="VOICEOVER" scope="All" />
   <tag name="CG" type="LOWERTHIRDS" scope="All" />
   <tag name="FLOAT" type="ADLIBPIX" scope="All" />
</tags>

<newsroomtypehandlernamemap>
   <tag name="CG" value="DSK" />
```

```
</newsroomtypehandlernamemap>

</inewstags>
```

when a graphic MOS object is parsed in Viz Mosart and found to be a CG command as defined in the <tag> node, Viz Mosart converts it to a Viz Mosart graphic item.
It then sets the **handler_name** attribute to *DSK*, based on the mapping defined in `<newsroomtypehandlernamemap>` node.
In this case, the graphic is redirected on the DSK channel of the configured graphic engine.

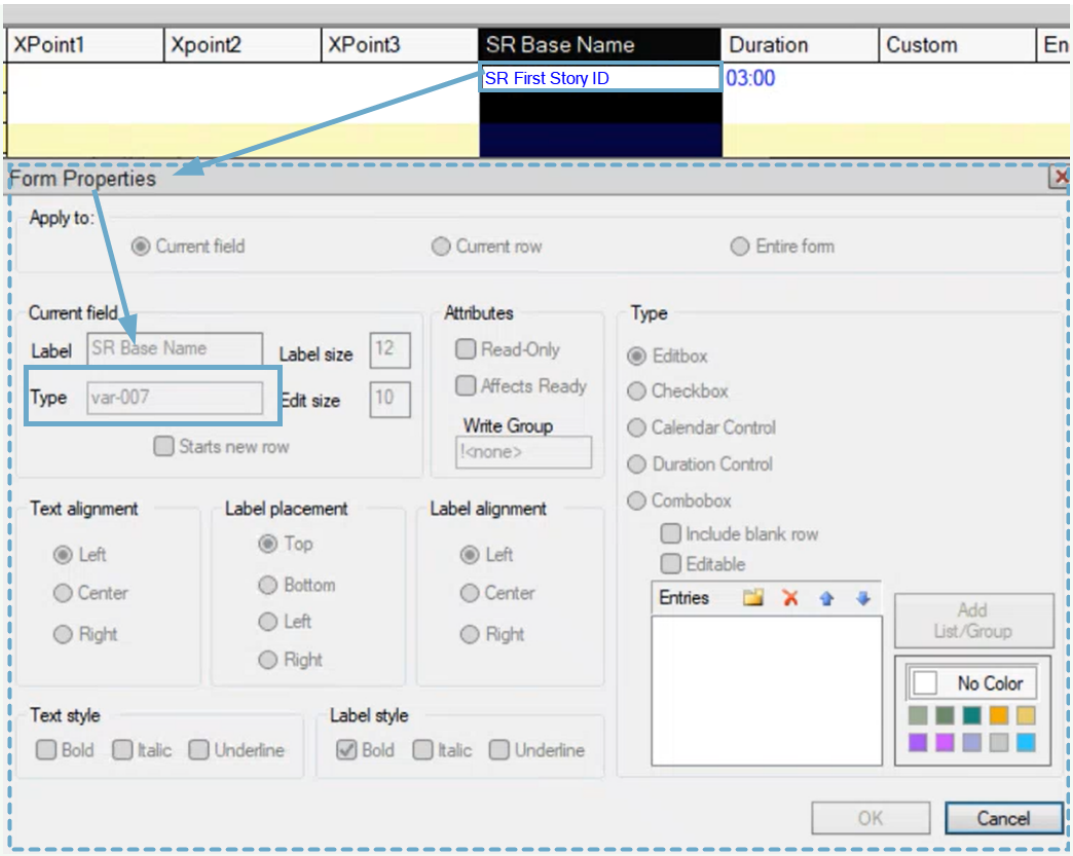
- **Omit text:** (iNews only) Omit text from iNEWS presenters area.
- **Properties:** (iNews only) iNEWS custom settings, see Overriding Default Field Names When Parsing iNEWS Stories below.
- **Ignore parenthesis:** (MOS-systems only) List of start and end parenthesis that encloses text ignored in Viz Mosart.
- **MOS Id mapping:** Enables Viz Mosart to recognize MOS elements from the NRCS and trigger specific Viz Mosart actions when encountered. Must not be changed by the user.
  See MOS ID Mapping below for a description of mapping MOS ID fields in the newsroomsettings.xml file.
- **MediaObject property mapping:** keywords specified during Viz Mosart installation for installation-specific behavior of Viz Mosart. Should not be changed by the user.

Story External Metadata

The NRCS can share story values in a container called **MosExternalMetadata** (MEM)**.** These values can then be translated to Viz Mosart content. The values are:

- **mostagname:** An XML element, often originating from a column name in the NRCS.

Tip: You can derive this NRCS-assigned name for example in *iNews*, by **Right-clicking > Form Properties**:



In this example, the user friendly *SR Base Name* is stored internally in iNews as *var-007*.

- **mos_value:** This is the text that appears in the XML element **mostagname** (set in the NRCS).
- **Mosart Action:**
  - **template_type:** Use a value from the NRCS to create a new Mosart item in the rundown story. The value from the NRCS can be a newsroom tag of TEXT item scope. Newsroom tags are configured in `newsroomsettings.xml`, (for example, *CAM*, *KAM*, *KAMERA*, *PKG*).
    You can also use the Viz Mosart type number, where for example, *0* is for Camera, *1* for Package. If the value from NRCS is *empty*, no item is created.
    Example:

    ```
    <mostag mostagname="var-1" mosart_action="template_type"
    format_type="String" />
    ```

    A new Viz Mosart item of type given in the NRCS column **var-1** is created.
  - **item_variable:** Uses the value from NRCS to set a field on the *first* primary item found in the story. Non-empty values overwrite any existing fields.
    Example:

    ```
    <mostag mostagname="var-2" mosart_action="item_variable"
    action_value="clip_hirespath" format_type="String" />
    ```

```
<mostag mostagname="var-3" mosart_action="item_variable"
action_value="output_preview" format_type="String" />
```

The fields **clip_hirespath** and **output_preview** are set on the first primary item found in the story, with the values taken from the columns **var-2** and **var-3** configured in NRCS.

- **item_duration:** Use the value from the NRCS column as the duration for an item or several items in the story.
  - If **action_value** is empty,
    - the duration is applied to the *first primary* item in the story.
  - If **action_value** contains the template type of a Viz Mosart item (given as a newsroomtag defined in the NRCS tags from Newsroomsettings or as a number (for example *CAM* or *0*) or a list of template types separated by comma,
    - the duration applies to *all* items of that type.
    For example: **action_value**=*CAM* or **action_value=***PKG, JINGLE* or **action_value=***0* or **action_value=***1,JINGLE*.
  - If **action_value** contains pairs `<type><separator><variant>` or a list of such pairs separated by a comma,
    - duration applies to those specific items in the story.
    For example: **action_value=CAM=***1* or **action_value=***CAM|1,PKG*).
    The `<separator>` can be "**=**", "**|**", "**:**" , "**-**" or "**;**".
    Use Format type to select if the value is given as seconds (integer), frames (integer) or as a time code.
- **template_variant:** Use the value from the NRCS to add or change the variant of the first primary item found in the story.
- **template_transition:** Use the value from the NRCS to add effect transitions to the first primary item found in the story.
  This also provides a good overview in the newsroom about which effects are used between the stories.
  For example:

```
<mostag mostagname="TranFromNcs" mos_value="SLIDE" format_type="String"
mosart_action="template_transition" action_value="MIX,24" />
```

  Use the value from the NRCS to add effect transitions to the *first primary* item found in the story.
  The effect transition is given in the format `<effect_name><separator><input>` , where `<separator>` can be white space, bar (**|**), comma(**,**), semicolon(**;**) or colon (**:**).
  The `<effect_name>` can be *EFFECT, MIX, WIPE* or their translations as configured in the **Keywords** section in Newsroomsettings.
  For example, if *EFFECT* is mapped as *EFFEKT* in the **Keywords** section, we can add the following effect to the Viz Mosart item: *Effekt 2*.
- **story_variable:** Use the value from the NRCS to set a field on all items (excluding prompters and some secondaries with type > 300) found in the story. Non-empty values do *not* overwrite existing fields.
- **story_duration:** Use the value from the NRCS as the planned duration for this story. Use **Format** type to select if the value is given as seconds (integer) or as a time code.

- **directtake_pre_story:** Executes the direct take template given in the **action_value** at the start of the story (in the switch from the previous to current story).
- **directtake_post_story:** Executes the direct take template given in the **action_value** at the end of the story (in the switch from the current to the next story).
- **guimarker_X:** Translates the set of **mostagname** and **mos_value** to a marker in the GUI, valid **action_values** are *blank* , *1* (green), 2 (blue) and 3 (red).
- **device_property:** Sends a key-value pair to the device driver. For example, for camera robots this can be used to adjust speed and camera positions.
- **story_endphrase:** Use the value from the NRCS to override the **endphrase** attribute for the *last* primary item in a Viz Mosart story.
- **segment_duration:** Use the value from the NRCS to set the duration of a group of stories.
- **words_per_minute** *:* Sets the read rate for a story.
- **back_time:** Sets the **backtime** attribute for a story (the exact time when a story must start in order for the show to remain on schedule.). This is used to set a next break time of the show.
  The **break** attribute must be set to *true***.**
- **cume_time:** Sets the **cumetime** attribute for a story - the amount of airtime required from the beginning of the show up to a certain point in the show in order for the show to remain on-schedule. This is used to set a next break duration of the show. The **break** attribute must be set to *true***.**
- **category:** Sets the **category** attribute for a story used to identify a story category such as sports, news, business etc.
- **field_value_X** *:* Overrides the field value of a Viz Mosart item with the value stipulated in the NRCS**.** This applies only to the first primary item in a story and the field must already exist for that item. For example, if the first primary item in a story is a Package with the following fields:

```
<fields>
    <field name="clip_description" value="" default="" fieldtype="TEXT"
keylist="" />
    <field name="clip_hirespath" value="" default="" fieldtype="TEXT"
keylist="" />
    <field name="metadata_lookuppath" value="" default="" fieldtype="TEXT"
 keylist="" />
</fields>
```

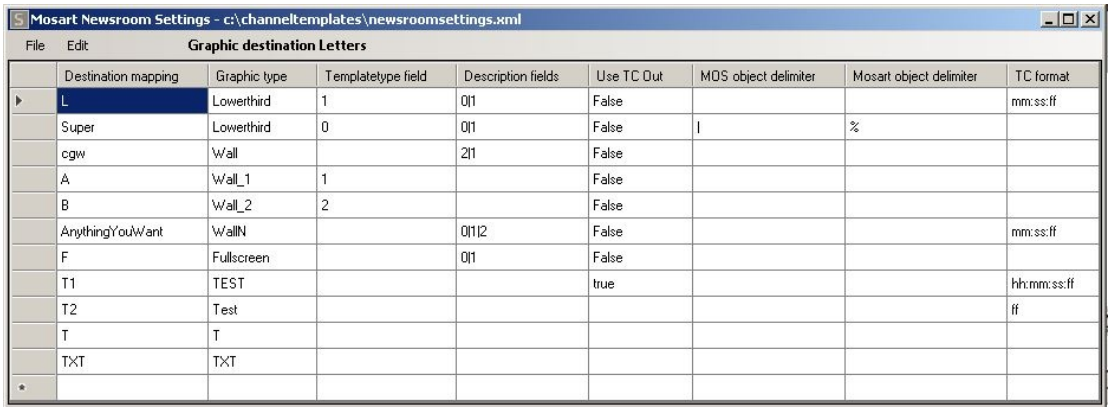
  and the following mappings are given in newsroom settings:

```
<mostag mostagname="ClipDescrCol" mosart_action="field_value_1"
action_value="clip_description" format_type="String" />
<mostag mostagname="ClipIdCol" mosart_action="field_value_2"
action_value="clip_hirespath=MyClip" mos_value="test"
format_type="String" />
```

  then the field is set to the value given in the NRCS in the column with id **ClipDescrCol** and the field **clip_hirespath** are set to the value **MyClip** if the value from column **ClipIdCol** in the NRCS is *test*.
- **asruninfo:** Sets the attribute **asruninfo** for all items in a Viz Mosart story. This is used to store specific information that is stored in the AsRunLog.

> **Info:** The attribute **action_value** for **<mostag>** node in newsroom setting can contain the *field name* or pairs of *field name=field value* separated by comma, colon, semi-colon,

Graphic Destination Letters



Destination mapping	Graphic type	Templatetype field	Description fields	Use TC Out	MOS object delimiter	Mosart object delimiter	TC format
L	Lowerthird	1	0|1	False			mm:ss:ff
Super	Lowerthird	0	0|1	False	|	%	
cgw	Wall		2|1	False			
A	Wall_1	1		False			
B	Wall_2	2		False			
AnythingYouWant	WallN		0|1|2	False			mm:ss:ff
F	Fullscreen		0|1	False			
T1	TEST			true			hh:mm:ss:ff
T2	Test						ff
T	T						
TXT	TXT						

Graphic destination letters identify output and behavior for graphics elements. Use them to parse a Vizrt MOS object and obtain a Viz Mosart object.

Properties

Each row contains the following columns:

- **Destination mapping:** The mapping defined in the Viz Mosart command from Viz super.
- **Graphic type:** Used to identify the output for the graphics elements, either on *predefined channels*:
  - **Fullscreen**: Full screen graphics for FULL output (channel)
  - **Lowerthird**: Overlays for DSK output
  - **Wall**: Overlays for WALL output
  - **WallN**: Overlays for outputs WALL_1, WALL_2, WALL_3 etc.
  - **Ticker**: Overlays for TICKER output

  or on *custom channels* (so any string can be set as graphic type).
- **Templatetypefield:** Index of the description field which is set as the TemplateType. (the value of the description field found at the specified index). The index is 0-based. Default: *empty***.**
  For example: If the **MosAbstract** is: `<mosAbstract>field1/field2/field3/field4/Mosart=S|00:04|00:10</mosAbstract>` , and you set `Templatetypefield=1` , then the `templatetype` attribute from the Viz Mosart object is set to `field2` .
- **Descriptionfields:** A list of indexes of the fields which should be shown in the description of a Mosart story item. These fields are shown in the Viz Mosart GUI separated by the character given in the **outputtxtdelimiter** setting. The indexes are 0-based and are separated using a pipe '|' character.
  For example: **descriptionfields=**"*0|1*". Default: *empty*.
- **Use TC Out:** Specifies if the duration should be used as **tc_out**.
- **Inputtxtdelimiter:** Vizrt MOS object delimiter. Default: '/' (slash)

- **Outputtxtdelimiter:** Viz Mosart object delimiter. Default: '/' (slash)
- **Mask:** Specifies the timecode format of the graphics starting time, end time or duration. Default: **mm:ss**. Example of formats: *hh:mm*, *hh:mm:ss*, *mm:ss:ff*, *mm:ss*, *ss:ff*, *ff*.

Mapping example:

```
<graphicdestinationletters>
  <graphicdestinationletter destinationmapping="L" type="Lowerthird"
templatetypefield="1" descriptionfields="0|1" use_tc_out="False" inputtxtdelimiter=""
 outputtxtdelimiter="" mask="mm:ss:ff"/>
  <graphicdestinationletter destinationmapping="Super" type="Lowerthird"
templatetypefield="0" descriptionfields="0|1" use_tc_out="False"
inputtxtdelimiter="|" outputtxtdelimiter="%" mask=""/>
  <graphicdestinationletter destinationmapping="cgw" type="Wall" templatetypefield=""
 descriptionfields="2|1" use_tc_out="False" inputtxtdelimiter=""
outputtxtdelimiter=""/>
  <graphicdestinationletter destinationmapping="A" type="Wall_1"
templatetypefield="1" use_tc_out="False" inputtxtdelimiter="" outputtxtdelimiter=""/>
  <graphicdestinationletter destinationmapping="B" type="Wall_2"
templatetypefield="2" use_tc_out="False" inputtxtdelimiter="" outputtxtdelimiter=""/>
  <graphicdestinationletter destinationmapping="AnythingYouWant" type="WallN"
templatetypefield="" descriptionfields="0|1|2" use_tc_out="False"
inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss:ff"/>
  <graphicdestinationletter destinationmapping="F" type="Fullscreen"
templatetypefield="" descriptionfields="0|1" use_tc_out="False" inputtxtdelimiter=""
outputtxtdelimiter=""/>
  <graphicdestinationletter destinationmapping="T1" type="TEST" use_tc_out="true"
mask="hh:mm:ss:ff"/>
  <graphicdestinationletter destinationmapping="T2" type="Test" mask="ff"/>
  <graphicdestinationletter destinationmapping="T" type="T"/>
  <graphicdestinationletter destinationmapping="TXT" type="TXT"/>
</graphicdestinationletters>
```

Walls

There are two ways to support a configurable number of walls:

1. By having a general rule in the graphics destination mappings where the graphics type must be *WallN***.** The **destionationmapping** attribute can be set to any string of alphanumeric characters.

   ```
   <graphicdestinationletter destinationmapping="Walls" type="WallN"
   templatetypefield="" descriptionfields="-1" use_tc_out="False"
   inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss"/>
   ```

   In this case, the Vizrt graphic objects must contain a Viz Mosart string command ending in a number (this provides the wall number).
   For example:
   `Mosart=cgw1|00:00|00:05`
   or
   `Mosart=something2|M|S`
   or

`Mosart=wall3|00:00|B`
or
`Mosart=W4|00:04|00:10`
and so on.
Given these examples, the destination for the first graphic is **WALL_1**, followed by **WALL_2**, **WALL_3** and **WALL_4**, respectively.

2. By not ending the Viz Mosart string destination mapping with a number.
For example:
`Mosart=B|00:00|00:05, Mosart=Z|00:00|00:05, Mosart=Wall|00:00|00:05, Mosart=Something|00:00|00:05` and so on.
In this case, the configuration in **newsroomsettings** must specify the wall number in the graphics type. For the Mosart string examples given above, the configuration can therefore be:

```
<graphicdestinationletter destinationmapping="B" type="Wall_2"
templatetypefield="" descriptionfields="-1" use_tc_out="False"
inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss"/>
<graphicdestinationletter destinationmapping="Z" type="Wall_3"
templatetypefield="" descriptionfields="-1" use_tc_out="False"
inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss" />
<graphicdestinationletter destinationmapping="Wall" type="Wall_4"
templatetypefield="" descriptionfields="-1" use_tc_out="False"
inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss"/>
<graphicdestinationletter destinationmapping="Something"
type="Wall_5"templatetypefield="" descriptionfields="-1" use_tc_out="False"
inputtxtdelimiter="" outputtxtdelimiter="" mask="mm:ss"/>
```

Overlays with
`Mosart=B|00:00|00:05`
are then redirected to **WALL_2**, those with `Mosart=Z|00:00|00:05` to **WALL_3** and so on.

As well as redirecting graphics to predefined destinations like *DSK, WALL, WALL_i, FULL*, the graphic destination letters rules can be used for any desired destination.
For example:

```
<graphicdestinationletter destinationmapping="T" type="TEST" templatetypefield=""
descriptionfields="-1" use_tc_out="False" inputtxtdelimiter="" outputtxtdelimiter=""
mask="mm:ss"/>
```

and a Vizrt graphic with Mosart string `T|00:00|00:06` sends the graphic for playout on the channel *TEST*.

6.8.2 Viz Mosart Object Structure

When building a graphic Viz Mosart object from a Vizrt MOS object, an XML structure is created:

```
<item type="100" slug="L3-Anna Smith|Vizrt reporter|Mosart=L|00:00|00:06" source="1"
index="100___2_5" idref="5" templatetype="TIMECODE-DSK" status="0" error="0" in="0"
```

```
dur="150" pin="0" pdur="150" rdur="0" externaleffect="" intimeline="true" date_0=""
accessory="False" static="false" endfrase="" mosid="PILOT" objid="121"
ismoselement="true" use_graphics_id="true" graphics_id="121" handler_name="DSK"
graphics_out_on="TIMECODE" description="(DSK) - L3-Anna Smith|Vizrt reporter|
Mosart=L|00:00|00:06" auto_continue="false">
  <fields>
    <field name="graphics_description" fieldtype="TEXT" value="L3-Anna Smith|Vizrt
reporter|Mosart=L|00:00|00:06"/>
    <field name="graphics_id" fieldtype="TEXT" value="121"/>
    <field name="tc_dur" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:06"/>
    <field name="continuecount" value="-1" fieldtype="TEXT"/>
    <field name="payloaduri" value="http://bgoemo:8177/dataelements/121/payload"
fieldtype="TEXT"/>
    <field name="thumbnailuri" value="http://bgoemo:8177/dataelements/121/thumb"
fieldtype="TEXT"/>
    <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:00"/>
  </fields>
</item>
```

The graphic *slug* is used to visualize the graphics in the Viz Mosart GUI **Assets** window or in the **Overlay Graphics** interface, for example. The slug is taken from **MosAbstract** found in the Vizrt MOS object and the fields delimited by the inputtxtdelimiter are replaced with outputtxtdelimiter.

Viz Mosart can receive several types of Vizrt MOS graphic objects from the NRCS, including a

- Vizrt MOS Item With Payload Data Nodes and a
- Vizrt MOS Item Without Payload Data Nodes.

Vizrt MOS Item With Payload Data Nodes

If a Vizrt MOS item contains `<mosPayload>` with data entries, then the **graphics_description** is built from these data nodes delimited by **outputtxtdelimiter**.

The property **inputtxtdelimiter** is used only to build the *slug*, not the description.

If **descriptionfields** is empty, then the **graphics_description** attribute from the Viz Mosart item (created from the Vizrt MOS item) is set to the *slug* value (created from **MosAbstract**, see above).

If the field is populated, the slug is set to the description created from data nodes entries where only the data nodes from the positions specified in the **descriptionfields** are added and separated by **outputtxtdelimiter**.

Example 1:

Using a Vizrt MOS item containing the following values:

```
<mosExternalMetadata>
  <mosScope>OBJECT</mosScope>
  <mosSchema>http://www.vizrt.com/mosObj/data</mosSchema>
  <mosPayload>
    <data>
      <entry name="data">
        <entry name="">
```

```
          <entry name="" type="widestring">Navn/Titel/Sted</entry>
        </entry>
        <entry name="1" description="Name">
          <entry name="1" description="Name" type="richtext" upper="true"
singleline="true" location="2/3/1/1">L3-Anna Smith</entry>
        </entry>
        <entry name="2" description="Title">
          <entry name="2" description="Title" type="richtext" upper="true"
singleline="true" location="2/2/2/1">Vizrt reporter</entry>
        </entry>
        <entry name="Mosart">
          <entry name="Mosart" type="widestring">Mosart=L|00:00|00:06 </entry>
        </entry>
      </entry>
    </data>
  </mosPayload>
</mosExternalMetadata>
```

apply the following mapping:

```
<graphicdestinationletters> <graphicdestinationletter destinationmapping="L"
type="Lowerthird" templatetypefield="" descriptionfields="" inputtxtdelimiter=""
outputtxtdelimiter="|" /> </graphicdestinationletters>
```

Note that **inputtxtdelimiter** is empty, so the default ‘ `/` ’ is used.

The result is the following Mosart item. Note the `slug` and `description` attributes and `graphics_description` field:

```
<item type="100" slug="L3-Anna Smith|Vizrt reporter|Mosart=L|00:00|00:06" source="1"
index="100___2_5" idref="5" templatetype="TIMECODE-DSK" status="0" error="0" in="0"
dur="150" pin="0" pdur="150" rdur="0" externaleffect="" intimeline="true" date_0=""
accessory="False" static="false" endfrase="" mosid="PILOT" objid="121"
ismoselement="true" use_graphics_id="true" graphics_id="121" handler_name="DSK"
graphics_out_on="TIMECODE" description="(DSK) - L3-Anna Smith|Vizrt reporter|
Mosart=L|00:00|00:06" auto_continue="false">
  <fields>
    <field name="graphics_description" fieldtype="TEXT" value="L3-Anna Smith|Vizrt
reporter|Mosart=L|00:00|00:06"/>
    <field name="graphics_id" fieldtype="TEXT" value="121"/>
    <field name="tc_dur" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:06"/>
    <field name="continuecount" value="-1" fieldtype="TEXT"/>
    <field name="payloaduri" value="http://bgoemo:8177/dataelements/121/payload"
fieldtype="TEXT"/>
    <field name="thumbnailuri" value="http://bgoemo:8177/dataelements/121/thumb"
fieldtype="TEXT"/>
    <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:00"/>
  </fields>
```

```
</item>
```

Example 2:

Using the same Vizrt MOS item as above, but with the following mapping:

```
<graphicdestinationletter destinationmapping="L" type="Lowerthird"
 templatetypefield="1" descriptionfields="0|1" outputtxtdelimiter="%" />
```

In the result below you can see that the delimiters are now '**%**' instead of '|'. Note that **templatetype** is set to the value found at index 1.

```
<item type="100" slug="L3-Anna Smith%Vizrt reporter" source="1" index="100___2_5"
idref="5" templatetype="Vizrt reporter" status="0" error="0" in="0" dur="150" pin="0"
 pdur="150" rdur="0" externaleffect="" intimeline="true" date_0="" accessory="False"
static="false" endfrase="" mosid="PILOT" objid="121" ismoselement="true"
use_graphics_id="true" graphics_id="121" handler_name="DSK"
graphics_out_on="TIMECODE" description="(DSK) - banner%Anna Smith"
auto_continue="false">
  <fields>
    <field name="graphics_description" fieldtype="TEXT" value="L3-Anna Smith%Vizrt
reporter" />
    ...
  </fields>
</item>
```

Vizrt MOS Item Without Payload Data Nodes

MosAbstract

If the Vizrt MOS item does not contain data nodes, but has **MosAbstract** set, then the slug and the description are built from **MosAbstract.**

In this case, **inputtxtdelimiter** is taken into consideration when building both the slug and description.

Similarly, if **descriptionfields** is empty, the **graphics_description** attribute from the Mosart item (created from the Vizrt MOS item) is set to the slug value (created from **MosAbstract**, as explained above).

Otherwise, the slug is set to the description built as explained above from MosAbstract where only the fields from the positions specified in **descriptionfields** are added and separated by **outputtxtdelimiter.**

Example 3:

Using the following Vizrt MOS Item:

```
<mosAbstract>00:00 | Super Tema + info | 26 | 1:Tema | 2:This is what the item is all
about |Mosart=S|M|00:04</mosAbstract>
```

apply the following mapping:

```
<graphicdestinationletters> <graphicdestinationletter destinationmapping="S"
type="Lowerthird" templatetypefield="" descriptionfields="1|3" inputtxtdelimiter="|"
outputtxtdelimiter="%" /> </graphicdestinationletters>"
```

The result is:

```
<item type="100" slug="Super Tema + info%1:Tema" source="1" index="100___2_2"
idref="2" templatetype="TIMECODE-DSK" status="1" error="0" in="0" dur="100" pin="0"
pdur="100" rdur="0" externaleffect="" intimeline="true" date_0="" accessory="False"
static="false" endfrase="" mosid="VIZ.NPRO.MOS" objid="1529718" ismoselement="true"
use_graphics_id="true" graphics_id="1529718" handler_name="DSK"
graphics_out_on="TIMECODE" description="(DSK) - Super Tema + info%1:Tema"
auto_continue="false">
  <fields>
    <field name="graphics_description" fieldtype="TEXT" value="Super Tema +
info%1:Tema"/>
    <field name="graphics_id" fieldtype="TEXT" value="1529718"/>
    <field name="tc_dur" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:04"/>
    <field name="continuecount" value="-1" fieldtype="TEXT"/>
    <field name="payloaduri" value="n/a" fieldtype="TEXT"/>
    <field name="thumbnailuri" value="n/a" fieldtype="TEXT"/>
  </fields>
</item>
```

6.8.3 Viz Mosart and Viz Content Pilot (VCP) MOS Item Integration

Vizrt MOS items do not contain any information about playout channel destination. By adding an optional **Mosart-part** in the **Vizrt data description**, properties for channel destination, automatic or manual play-out and timing information can be extracted from the MOS item.
This section presents the Vizrt MOS item metadata required to allow Viz Mosart to separate play-out channel and extract timing information.

Basic requirements

Viz Mosart requires details on whether the graphics element is a lower third, a graphics played out on a video wall engine, or a full screen graphics. Any graphic requires the following properties:

- **Destination**: lower third, video wall or full-screen graphics.
- **For lower thirds/video walls**:
  - Play-out: manual or automatic
  - For automatic play-out: in time
  - Duration/out time or the special durations background end, story end or open end.
- **Mosart item** (optional): Stored at the end of the description of the data element or as a specified field in the **mosExternalMetadata** section of the MOS item and is formatted as follows:
  Mosart={destination}|{in behaviour}|{out behaviour}|{X or F or L}
  where:

- **{destination}** is an alphanumeric string specifying the destination of the graphic. Suggested values are **L**=lower third, **W**=wall, **F**=full screen graphics.
- **{in behaviour}** is
  - either **M** for manual play-out
  - or a time code formatted as **mm:ss** (minutes, seconds) as in time.
- **{out behaviour}** is
  - either a time code formatted as **mm:ss** (minutes, seconds) as duration
  - or a single character field with values **B**, **S** or **O** where **B**=background end, **S**=story end and **O**=open end.
- **X** is used only for lower thirds to ignore “Replace TakeOut Logic” meaning that when taking and taking out the item, the replace logic is ignored if the value is set on the item.
- **L** is used to designate that this graphic should be treated as a “Locator,” meaning it is automatically attached to the crosspoint used in the initial template for the story. When that crosspoint is taken from a Viz Mosart template, whether as a switcher crosspoint or as a keyed crosspoint in a DVE box, the graphics are also re-taken.
  Not to be confused with {destination} letter “L” normally used for lower third graphics.
- **F** is used only for lower thirds to force “Replace TakeOut Logic” for graphics coming from different templates with same layers (when used, it practically ignores the “O” state for a layer).
  Not to be confused with {destination} letter “F” normally used for full screen graphics.

The rules which define the output and behavior for graphics elements can be configured in Graphic Destination Letters.

Field Example:

The following table contains examples of the **Mosart= part** of the Vizrt data element description in the value **cg_gordon_brown/prime minister/Mosart=L|00:02|00:05**. The data element is played out as a lower third, automatically in at two seconds with five seconds duration.

L|00:02|00:05	Lower third, automatically in at 2 seconds with 5 seconds duration
L|00:02|S	Lower third, automatically in at 2 seconds with duration equal to the length of the story (taken out at the switch from one story to the next)
L|00:00|O	Lower third, automatically in at 0 seconds, never taken out (except for when replaced by another item or manually from the operator)
L|M|00:05	Lower third, manual play-out, duration 5 seconds
L|00:01|B|L	Lower third, automatically in at 1 second with background end, designated as a Locator
L|00:00|00:10|X	Lower third, automatically in at 0 seconds with 10 seconds duration and Replace TakeOut Logic is ignored
W|00:00|B	Video wall, automatically in at 0 seconds, out when switching from one story element to another

F	Full screen graphics (note that this letter is usually used for full screen graphics, but any combination of alphanumeric characters can be used to identify a fullscreengraphic)

If no fields are given, the element defaults to a full screen graphics element.

VCP template requirements

The Mosart field is supported when embedded as a part of the VCP data element description or as a dedicated field in the **mosExternalMetadata** section.

Storing the Mosart field in the VCP data element description

When saving the VCP data element, the **Mosart** field shall be appended to the stored database description (the name of the data element).

Storing the Mosart field in the mosExternalMetadata section of the VCP data element

This method is only supported if the **mosExternalMetadata** data section is enabled for the VCP Template Filler ActiveX. See Appendix A for configuring the ActiveX to include this section in the MOS object.

The scene needs a **ControlObject.** The Mosart field should be stored in a hidden text field in the template, and the text field shall use **Mosart** in the **ControlObjectName** property. The Mosart field can either be included in the scene or added manually after importing the scene to the Viz Template Wizard.

VB script Example

The following VB script can be used in a Vizrt Template Wizard template and returns a properly formatted Mosart field string:

```
'' Method that returns a properly formatted Mosart description
Function GetMosartDescriptionPart(AType, AVariant, AOperation, ATCIn, ATCOut)
  'Template variant
  Dim FType, FVariant, FDefaultVariant, FDefaultOperation
  Select Case AType
    Case "LOWER"
       FType = "L" 'LOWER (lower third, over shoulder gfx)
       If AVariant = "" Then
         AVariant = "AUTOOUT"
       End If
    Case "WALL"
       FType = "W" 'LOWER (lower third, over shoulder gfx)
       If AVariant = "" Then
         AVariant = "OPENEND"
       End If
    Case Else
       'FType = "F" 'FULL (fullscreen)
       GetMosartDescriptionPart = "Mosart=F"
       Exit Function
  End Select
```

```
  'Take out logic (Only applies for lower thirds)
  FVariant = TranslateVariant(AVariant)

  'Operation selection
  Select Case AOperation
    Case "MANUAL"
      FOperation = "M" 'FULL
      FTCIn = FOperation
      If (FVariant="A") Then
        FTCOut = ATCOut
      Else
        FTCOut = FVariant
      End If
    case else 'case "AUTO+"
      FOperation = "A" 'AUTO+
      FTCIn = ATCIn
      If (FVariant="A") Then
        FTCOut = ATCOut
      Else
        FTCOut = FVariant
      End If
  end select

  GetMosartDescriptionPart = ToMosartProperty(FType, FTCIn, FTCOut)
End Function

Function ToMosartProperty(AType, ATCIn, ATCOut)
  splitChar = "|"
  Dim vals
  'vals = vbNewLine
  If Not AType = "" Then
    vals = vals & "Mosart="
    vals = vals & AType
    If Not ATCIn = "" Then
      vals = vals & splitChar & ATCIn
      If Not ATCOut = "" Then
        vals = vals & splitChar & ATCOut
      End If
    End If
  End If
  ToMosartProperty = vals
End Function

Function TranslateVariant(AVariant)
  select case AVariant
    case "OPENEND"
      TranslateVariant = "O" 'OPENEND
    case "STORYEND"
      TranslateVariant = "S" 'STORYEND
    case "BACKGROUNDEND"
      TranslateVariant = "B" 'BACKGROUNDEND
    case else 'case "AUTOOUT"
```

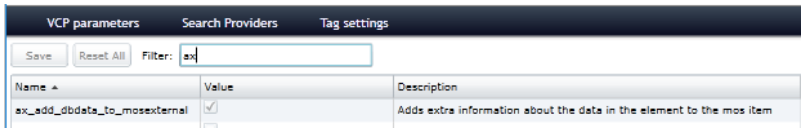
```
      TranslateVariant = "A" 'AUTOOUT
  end select
End Function
```

Viz Pilot plugin Prerequisites

To enable the **mosExternalMetadata** section in the Viz Pilot News (Active X) plugin, the system administrator must also enable a field in the VCP parameters list in the Viz Pilot database.

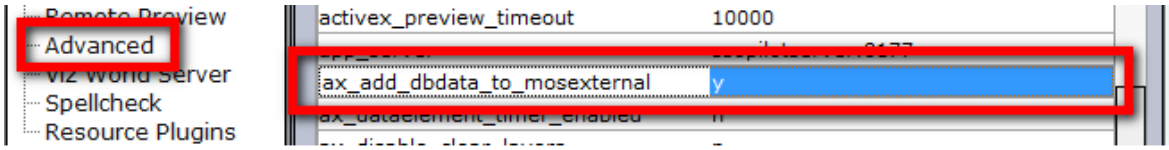
If you Viz Pilot 5.7 or later, this setting is available in the Viz Pilot Data Server settings page found in:

http://[hostname of PDS]:8177/settings#Params



In the **Value** column, the checkbox should be checked (set to true).

If you have a Viz Pilot version older than 5.7 this is easily achieved by opening the Preferences editor in the **Viz Content Pilot** (**Options → Preferences → Advanced**), by adding or editing the field called **ax_add_dbdata_to_mosexternal** setting the value to **y**.



6.8.4 MOS ID Mapping

A **MOS Id** mapping can be performed in the `newsroomsettings.xml` file, as follows:

```
<mosids>
    <tag name="" value="" fieldMapping="" appendContent="" embeddedMosartItem=""
 keywords="" />
</mosids>
```

Where:

- **name:** The MOS ID that identifies the MOS elements coming from the NRCS**.**
- **value:** The mapping ID used to identify a mapping for MOS element fields to be parsed. The MOS element's fields to be parsed are defined using `<mosmaps> (/mmConstants/xmltemplates/mosmaps/mosmap/).`
- **fieldMapping:** Set to *true* if field mapping should apply.
- **appendContent:** Set to *true* if the mapping should be appended to the Viz Mosart element item.

- **embeddedMosartItem:** Set to *true* if the MOS element has a Viz Mosart command embedded.
- **keywords:** Only the MOS parsing of Vizrt objects uses this column. It contains a comma-separated list of device-specific values.
  - `novideo` : Adding this excludes video clip information from Vizrt Video MOS objects, and only uses the lower thirds of the MOS item. The video ID can then be given from any other MOS object, or a story field. For example the video id column in iNEWS**.**
  - `noedit` : Disables the option of editing a graphic with the associated MOS ID from within the Mosart GUI. The *Edit* option in the context menu and double-click within the Assets window will be disabled.
  - `nodefaultitem:` Adding this excludes Vizrt full-frame graphics if no default item is found (when missing Viz Mosart template).
  - `IgnoreTemplateSet` (or `IgnoreTemplateSet=true`): When added to a MOS ID named `"MOSART"` , the **templateset** attribute from a MOS object coming from the NRCS is ignored.
    For example:
    `<tag name="MOSART" value="" appendContent="false" fieldMapping="false" keywords="IgnoreTemplateSet=true"/>`

6.8.5 Additional Newsroom Settings Examples

Defining Default Values for Lower Thirds

You can set default values for parsing of lower third objects. Usually these values are set from the objects received from the NRCS. Altering these values is usually part of a customization task.

- In `newsroomsettings.xml` add the default values to the *mmConstants/inewstags/properties* element:

```
<mmConstants><inewstags>
    <properties>
        <item name = "DefaultLowerThirdDuration" value = "125"/>
         <item name = "DefaultLowerThirdHandlerName" value = "DSK"/>
         <item name = "DefaultLowerThirdOutBehaviour" value = "TIMECODE"/>
         <item name = "DefaultLowerThirdOwnerMedia" value = "MEDIA"/>
    </properties>
</inewstags></mmConstants>
```

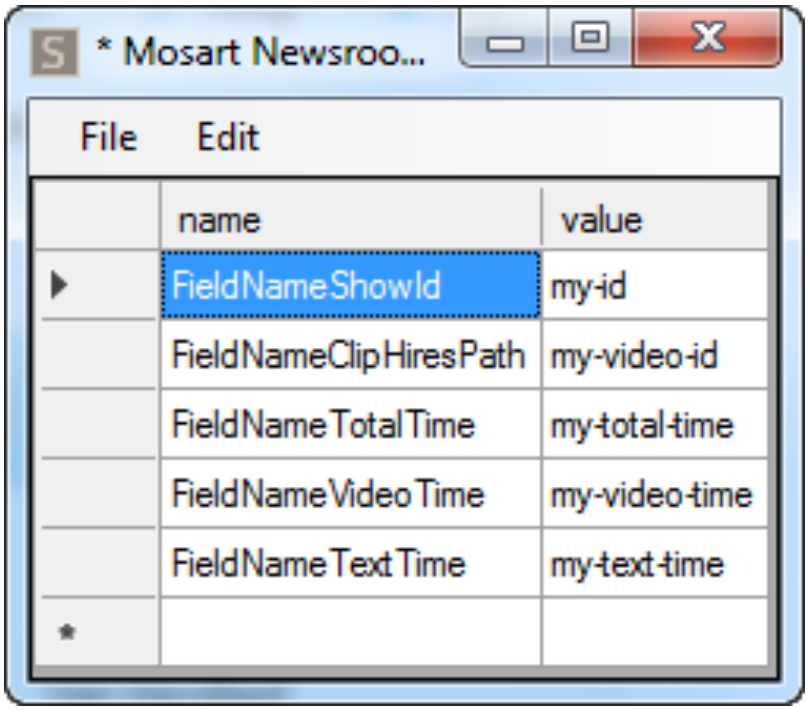
Lower Third Properties

Property	Description	Default value
DefaultLower ThirdDuratio n	The default duration of a TIMECODE lower third, in frames	125

Property	Description	Default value
DefaultLower ThirdHandler Name	The default graphics destination / handler	DSK
DefaultLower ThirdOutBeh aviour	The default out behaviour	TIMECODE
DefaultLower ThirdOwner Media	The default owner attribute for secondary NRCS objects	MEDIA

iNEWS: Overriding Default Field Names when Parsing Stories

When parsing the iNEWS NSML, the system extracts various field values using default field names. These defaults can be overridden by setting new values in **Newsroom Settings > Edit > Properties**.



Field name	iNEWS default field name	Description
FieldNameShow Id	id	The house ID of the show. Used for as-run logging.

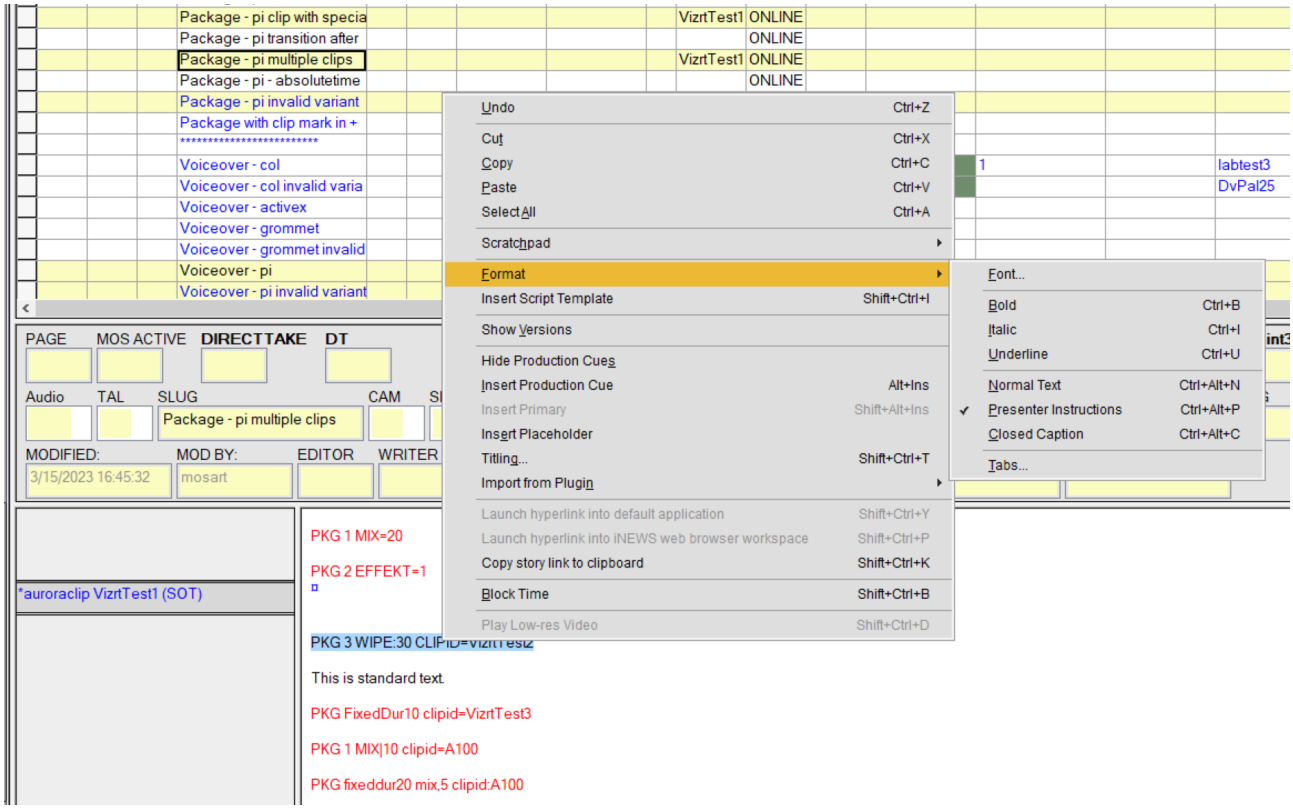
Field name	iNEWS default field name	Description
FieldNameClipHiresPath	video-id	The video ID column. The field is appended to the first primary element in the story.
FieldNameTotalTime	total-time	The total planned duration for the story.
FieldNameVideoTime	tape-time	The video time of the story.
FieldNameTextTime	text-time	The iNEWS-calculated text time of the story. If the total time is not available, the value is calculated using this value is added to video time.

iNEWS: Presenter Instructions

You can use Mosart commands as presenter instructions in iNEWS for item types CAMERA, PACKAGE, VOICEOVER, ENDFRASE, ATTACHA, CLIPSTART, FULLSCREENGRAPHICS, DVE, TELEPHONEINTERVIEW, BREAK, JINGLE, LIVE, ADLIBPIX, CLIPHIRESPATH and ACCESSORIES.

A presenter instruction (*red text*) command in iNews is a text command written in the story script. The journalist designates text as a presenter instruction by selecting the text, right-clicking and selecting **Format > Presenter Instruction** (or shortcut **Ctrl+Alt+P**). The selected text appears in red indicating it is not standard story text.

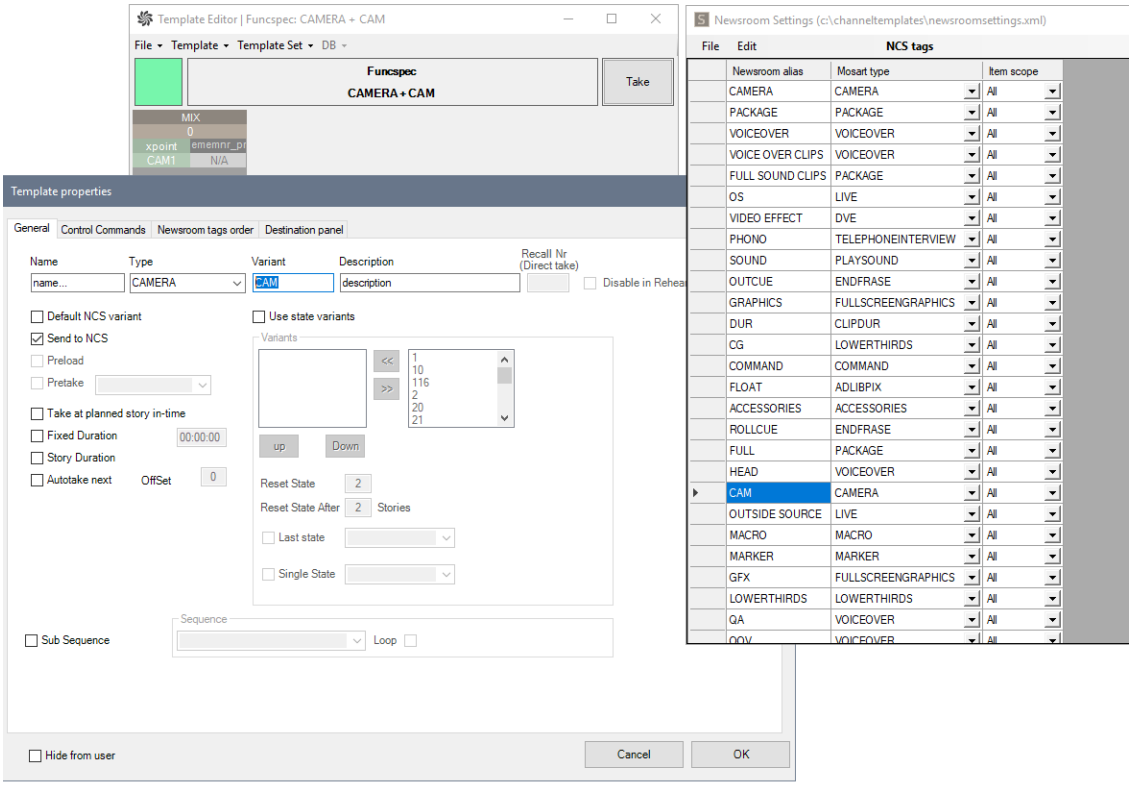
Copyright © 2026 Vizrt



To set up presenter text

Before working with presenter text in iNEWS, so one-off setups must first be made in Viz Mosart.

1. In Template Editor make sure you have template types defined for *all* (except for *Cliphirespath* and *Endfrase* which are not template types) the item types listed above.
   a. In Viz Mosart's AV Automation console, from the menu bar select **Devices > Template editor**.
   b. You will use presenter commands without specifying a template variant.
   You need to create a template variant with the same name as the newsroom alias, as shown below.



2. If not already defined, you must first add newsroom aliases for each of the following item types: Camera, Package, Voiceover, Live, Dve, Telephoneinterview, Break, Jingle, Adlibpix, Accessories, Cliphirespath, Endfrase.

> **Info**: There can be several newsroom aliases for the same item type.
> You can use any of them for commands in your iNEWS story for defining a Mosart story item in a Mosart template.

   a. In Viz Mosart's Manus Administrator console, type *ns*.
   b. From the menu bar, select **Edit > NCS tags**.
   c. Create mappings for any of the item types above that do not yet have an alias.

To work with presenter text

The following general commands can be used for CAMERA, LIVE, FULLSCREENGRAPHICS, DVE, TELEPHONEINTERVIEW, BREAK, JINGLE:

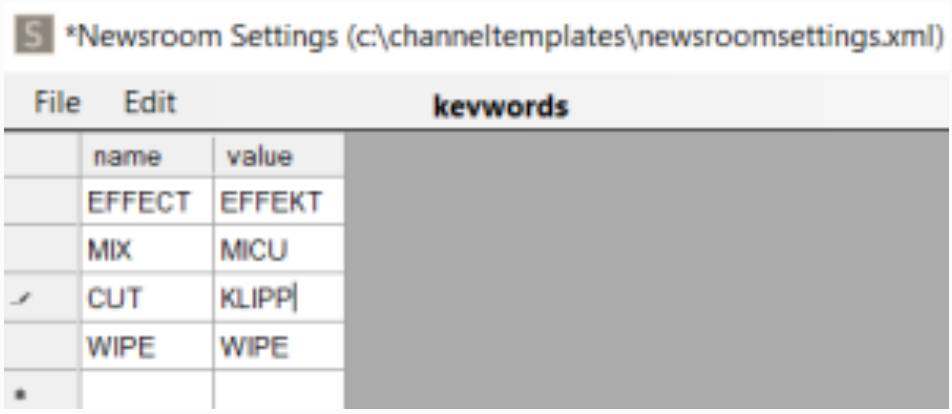
```
<ncs-alias> <variant> <transition><sep><value>
```

- where
- **<ncs-alias>** is a Newsroom alias for the supported Mosart template types as defined in **Mosart Manus Administrator > Edit > NCS tags**:
  For example, the NcsAlias could be *cam* of type CAMERA.

- **<variant>** is a template variant defined in the Template Editor (step 1 above) for the given type.
  If no variant is provided in the command, **<ncs-alias>** is used as template variant, which is why you need to have a template defined in Template Editor with variant **<ncs-alias>**.
- **<transition>** is the video transition when switching from one story item to the next in a show.
  Can be by default MIX, WIPE or EFFECT. The value for *mix* and *wipe* is the rate of the transition (in frames), while for the *effect* this is the effect number *(Physical Input)* configured in **AV Automation > Devices > A/V setup**.

Info: For **<transition>**

- Names are not case sensitive.
- You can use a separator equal (=), bar(|) or colon (:).
- Default Viz Mosart naming can be overridden with preferred names by navigating to the Mosart Manus Administrator console, typing `ns` followed by **ENTER**. Then **Edit > keywords** and mapping a new *value* to the default *name*.



This could, for example, be a local language text.

To work with clips

You can specify the clip id for clip types (PACKAGE, VOICEOVER or ADLIBPIX), enabling multiple clips to be added to a single story.

The command format is:

```
<ncs-alias> <variant> <transition><sep><value> clipid=<clipname>
```

where:

- **<ncs-alias>** is a Newsroom alias for the supported clip Mosart template types (PACKAGE, VOICEOVER, ADLIBPIX) in **Mosart Manus Administrator > Edit > NCS tags**.
  For example, the NcsAlias could be *pkg* of type PACKAGE.
- **clipid** assigns a clip to the Mosart item created with this command.

Info: For ***<clipname>***

- This is the name of a clip residing on the MAM system controlled by Viz Mosart.

- clip name must not contain spaces.
- This parameter is useful when defining multiple clips in the same story. The first clip can get the clip id from the VID-ID column, the other clips using this option on the command line.

- If clipid is given for the first clip in the story, this overrides the value given in the VID-ID column.
- **<transition>**: same as above.
- **@** character. The absolute time field will be added to the element indicating that the element will be prepared and taken on at absolute time from midnight.

To attach a clip to last Viz Mosart item

The type CLIPHIRESPATH can also specify a clip id that will be attached to the last Mosart item.

The full command is

```
<ncs-alias> <clipname>
```

where

- **<ncs-alias>** is a Newsroom alias of type CLIPHIRESPATH as defined in **Mosart Manus Administrator > Edit > NCS tags**.

To work with accessories

The type ACCESSORIES can be used with manual or time-controlled takes.

The command format is

- For auto accessory with timecode in and/or timecode out relative to the start of the story item:

```
<ncs-alias> <variant> tc-in=mm:ss tc-out=mm:ss
```

- For manual accessory which can be taken manually or executed at the absolute time specified by @:

```
<ncs-alias> <variant>  @hh:mm:ss|@hh:mm
```

where:

- **<ncs-alias>** is a Newsroom alias of type ACCESSORIES as defined in **Mosart Manus Administrator > Edit > NCS tags**.
- **tc-in** is the time when the accessory is taken in, relative to a story item start.

Info:
- If no **tc_in** is given, the accessory is considered *manual*.
- Other keywords for **tc-in** are **tcin**, **tc_in** or simply **in**

- **tc-out** is given to calculate the duration for an accessory item.

Info: Other keywords for **tc-out** are **tcout**, **tc_out** or simply **ou**

- **@** character. Same as above.

To work with an end phrase

This command attaches the text given in the command to the story item preceding this command.

The command format is

```
<ncs-alias>=some text
```

- The text is displayed in the **Program** area in Viz Mosart UI if *Show end phrase* is enabled for this template type, from the menu
  Viz Mosart UI > Tools > General settings > Preview/Program window > End phrase bar > Show end phrase <type>.

6.9 iNEWS Timer

6.9.1 Working with the iNEWS Timer

The iNEWS Timer provides accurate timing of the production in iNEWS, and is used to synchronize the current Viz Mosart on-air story in the GUI, with its corresponding entry in iNEWS.

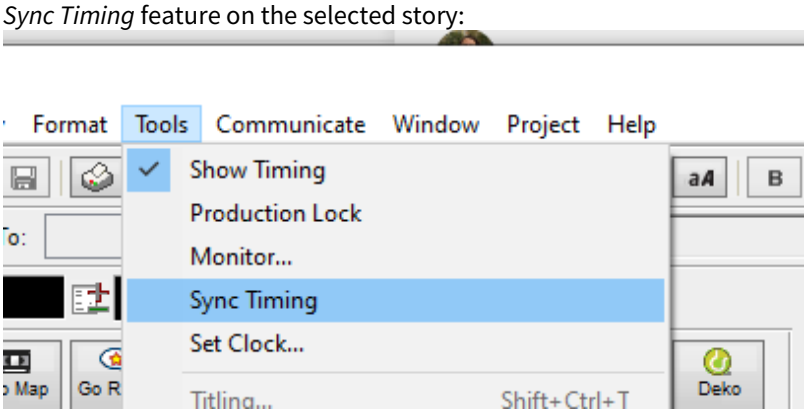
Connection

Connected to a local iNEWS client and the remote Mosart server, this application sends a shortcut to the iNEWS client each time an item is stepped through in the Viz Mosart story.

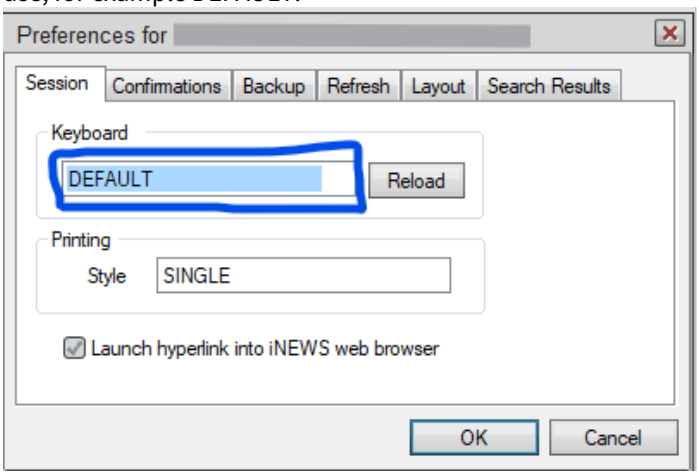
To set up an iNEWS Timer

> **Info**: iNEWS Auto Timer application must be started with Administrator privileges, to be able to send shortcuts to the iNEWS client.

1. First run: From the Viz Mosart installer directory, run VizMosartINEWSTimer-nnn.msi
2. Start iNEWS Timer from the desktop shortcut.
3. After the iNEWS Timer starts, a console Viz Mosart - iNEWS Auto Timer opens.
   Type *settings* in the console, to enable configuration.
4. Make the following configurations:
   a. **ApplicationProcessname**: Name of your local iNEWS Client process. Default: *ANWS*.
   b. **KeyStrokeHome**: iNews client shortcut to reach top of rundown. Default: *^{HOME}*.
   c. **KeyStrokeTimeMarker**: iNEWS client shortcut to be executed at each story. Default: *{F12}*.
   d. **MMServerHost**: Mosart Server hostname/IP. Default: *none*.
   e. **MMServerHostB**: Mosart Backup host. Default: *none*.
   f. **ShowKeyStrokes**: Display sent keystrokes. Default: *false*.
   g. **ShowWindow**: Determines how the iNEWS Client window will behave when the **KeyStrokeTimeMarker** is sent. Default: *5*.
   0: Simulates an Alt+Tab switch to the iNEWS Client window
   1: Maximizes the iNEWS Client window, then brings it to front
   2: Brings the iNEWS Client to front without changing window state
   3: Same as case 1
   default Same as case 0
5. In the iNEWS client, there are some steps needed to configure an iNEWS keyboard shortcut for triggering the *Sync Timing* feature on the selected story:



a. In the iNEWS client, open **Tools > Options > Preferences** menu, to determine which keyboard is in use, for example DEFAULT:



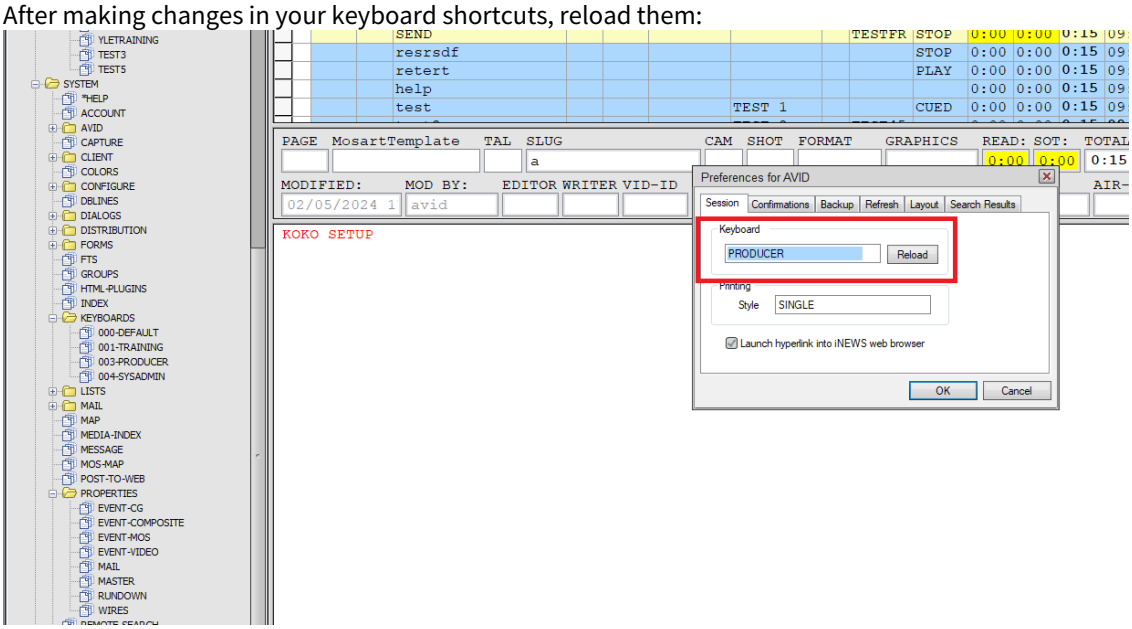
b. Open the Queue SYSTEM.KEYBOARD.XXXX where XXXX is the keyboard is in use (for example, *DEFAULT*).

c. Add the *F12 TIME MARK* shortcut for triggering **Tools > Sync Timing**.
(If something other than *F12* is configured as **KeyStrokeTimeMarker**, adapt this to suit that setting).
{alt t}: Selecting **Tools** top menu tab
{shift s}: Selecting **Sync Timing**

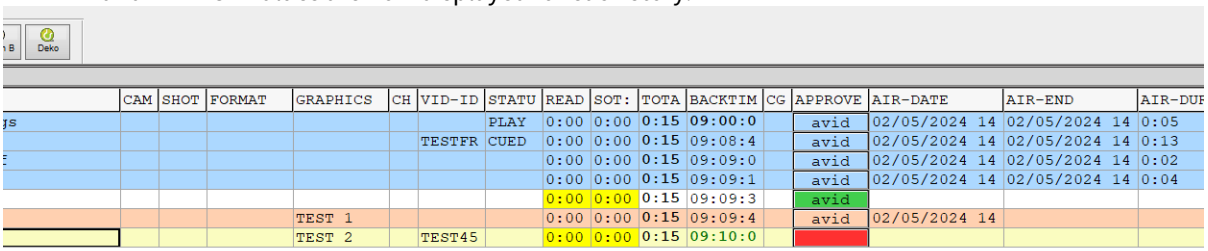
```
;F12 Time Mark
```

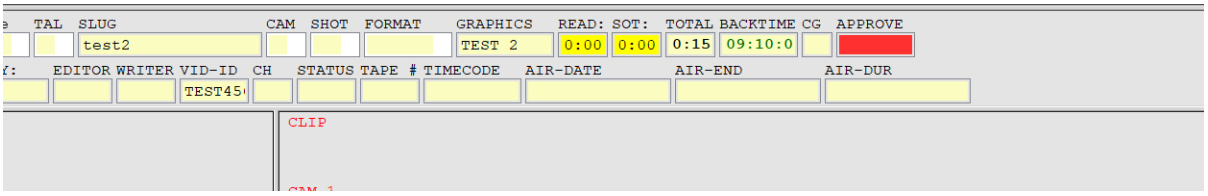
```
@{f12}~{alt t}{shift s}
```

d. After making changes in your keyboard shortcuts, reload them:



6. AIR-END and AIR-DUR values are now displayed for each story:





Command Console Commands

From the iNEWS Auto Timer command console you can enter the following commands:

- help
- SETTINGS: Open the Settings window
- SEND [keystroke]: Send a keystroke
- SENDHOME: Sends the *currenthome* message
- SHOWWINDOW [int]: Window position
- SHOWKEYSTROKES [y]: Shows the sending keystrokes
- PRCNAME [ProcessName]: Details on running process.

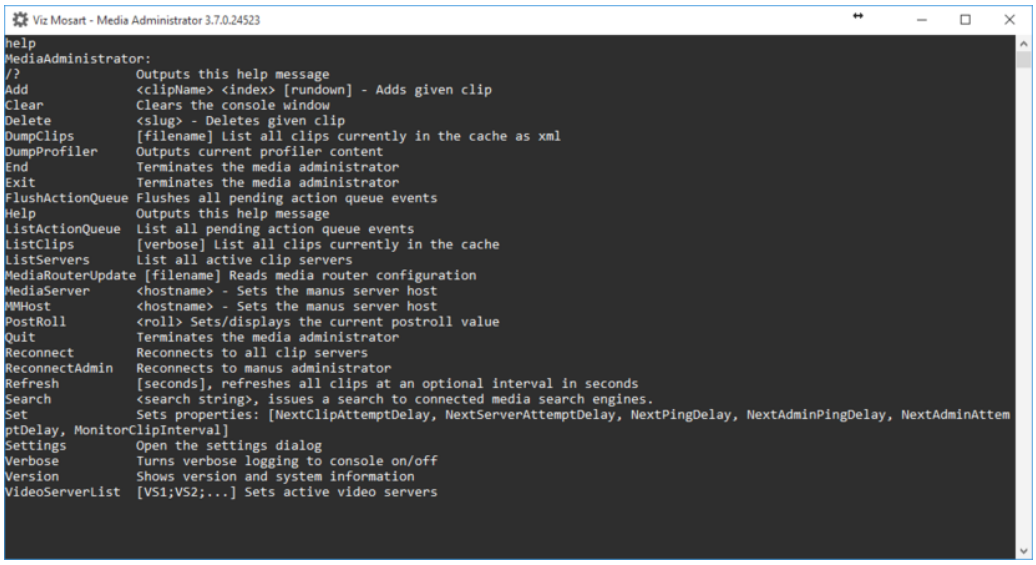
See also: Avid iNews Web Service for Status Feedback.

7 Media Administrator

Media Administrator handles the database connection to your video servers. It is responsible for reporting the clip status for all the video server elements in the current rundown, and for searching the video server for clips within the Viz Mosart GUI.

- For the supported MAM servers, refer to Video Server, MAM and Database Connection Strings.

Media Administrator runs as a console application and should be kept running at all times as long as its services are required.



Media Administrator receives the video server configuration (which servers are active) in the following circumstances:

- when the **Media Administrator** is started.
- when the **Manus Administrator** is activated (from *idle* to *active transition*).
- when the **AV Automation** changes the current setup.

To open the **Settings XML Editor**, type `settings` in **Media Administrator** followed by **ENTER**.

> **Info:** The application must be restarted to apply any saved changes to Settings.

This section contains the following topics:

- Media Administrator Properties Editor
- Media Administrator Commands

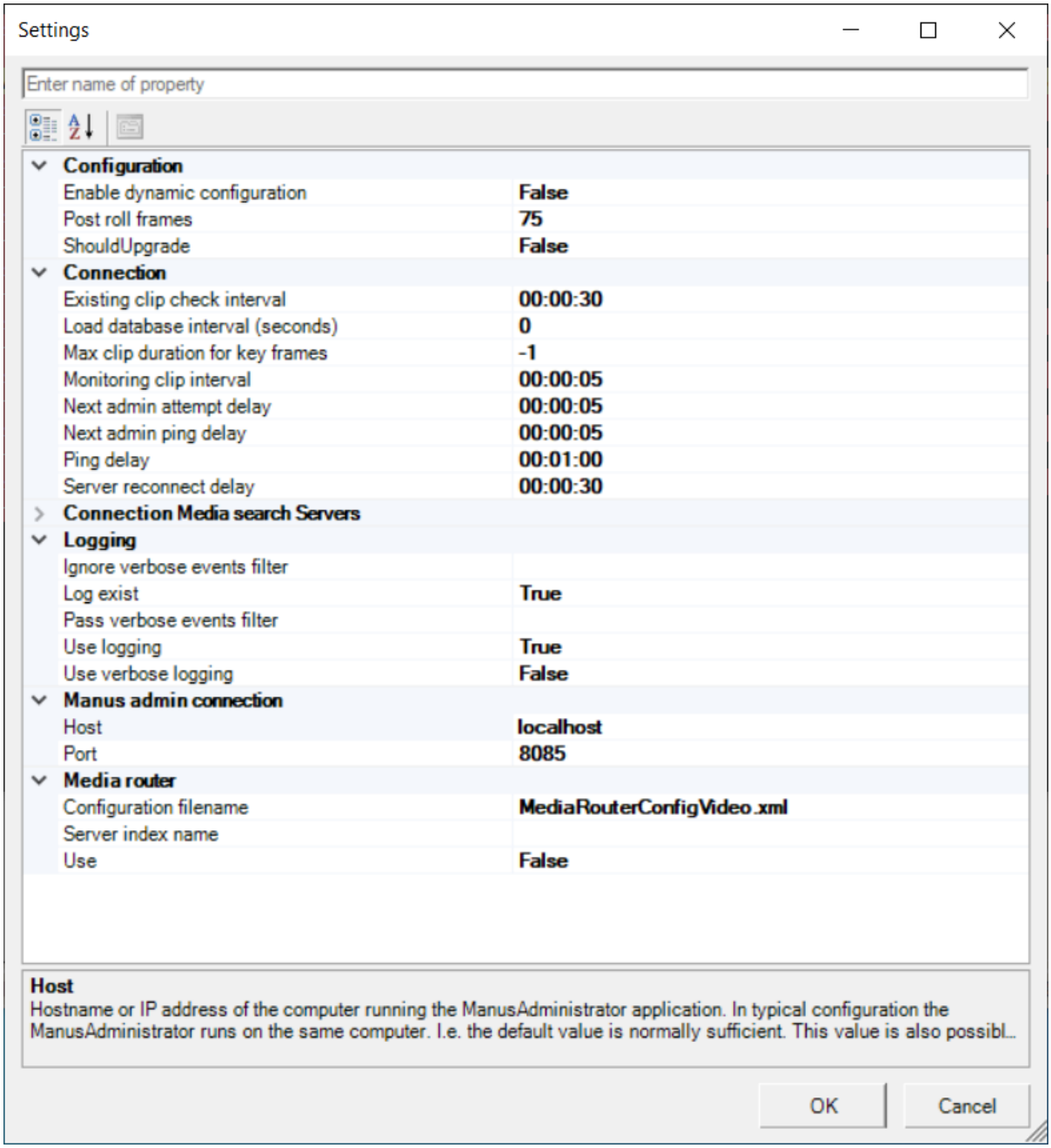
7.1 Media Administrator Properties Editor

Media Administrator is configured through the **Properties XML Editor**, which is opened by typing `settings` in the console window.

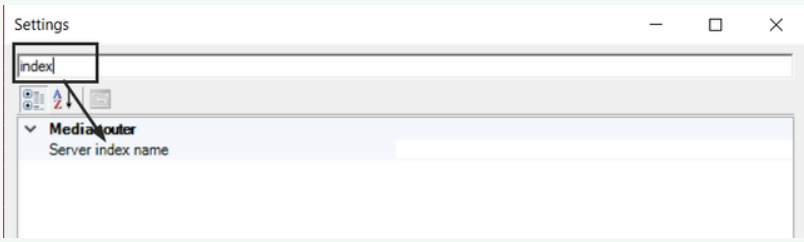
> ⚠ **Note:** Once you have made any changes, you must restart the Media Administrator for changes to apply.

- Sections of the Mosart Media Administrator Properties File
  - Configuration
  - Connection
  - Connection Media Search Servers
  - Logging
  - Manus Admin Connection
  - Media Router
- Dynamic Video Server Configuration
  - Enabling Dynamic Configuration

Copyright © 2026 Vizrt



Tip: You can locate a specific setting from the Search box:



7.1.1 Sections of the Mosart Media Administrator Properties File

Configuration

- **Enable dynamic configuration** (EnableDynamicConfiguration):
  - When *False* (default), *Media Administrator* connects to all servers configured (see Connection Media Search Servers, **Video clip Server <n>** below).
  - When True, it connects only to
    - Servers that are part of the salvo selected in *AV Automation* (see AV Automation, **Switch** button)
      and
    - Servers which have Static=*True* in the connection string (see Connection Media Search Servers, **Video clip Server <n>** below).

> **Note**: For further details, see section Dynamic Video Server Configuration below.

- **Post roll frames** (PostRollFrames): Subtracts value from the actual clip length when sending clip info back to the **Manus Administrator**. Default: *75* (frames).
- **ShouldUpgrade**: For Vizrt Support use only, must be set to *False*. Default: *False***.**

Connection

- **Existing clip check interval** (ExistingClipCheckInterval): Delay in seconds when to automatically verify the presence of non-existing clips. Note that most video servers support asynchronous notifications, making this functionality unnecessary. Default: *00:00:30.*
- **Max clip duration for key frames** (MaxClipDurationForKeyFrames): Disables showing keyframe markers for video files greater than this value in frames. Default: *-1* .
- Monitoring clip interval (MonitoringClipInterval): Default interval for monitoring clips. A clip is polled for changes as long as any of its attributes changes. Default: *00:00:05.*
- **Next admin attempt delay** (NextAdminAttemptDelay): Interval for retrying reconnecting to the Manus Administrator. Default: *00:00:05* (hh:mm:ss).
- Next admin ping delay: Interval for sending heartbeat to the Manus Administrator. Default: *00:00:05 (hh:mm:ss).*
- **Ping delay** (PingDelay): Interval between command requests (heartbeats) that Media Administrator sends to the video server. Pinging is used to detect whether the connected video servers are valid. Default: *00:01:00.*
  For some video servers, ping can be disabled using the DisableHeartbeat property inside the connection string. See, for example, Configuration File Properties - VDCP > DisableHeartbeat.
- **Server reconnect delay** (ServerReconnectDelay): Delay in seconds when to try to reconnect to a video server. For example, if connecting to a video server is unsuccessful, this delay determines the time to wait before a new connection attempt is made. Default: *00:00:30.*

Connection Media Search Servers

- **Video clip Server <n>** (VideoClipServer <n>): Connection string to a media server (a video server or MAM system) <n>.
  For details, see Video Server, MAM and Database Connection Strings.
  A maximum of six media servers may be connected. In most cases more than one server will be connected. These connection strings all share a common format of a semicolon-separated list of pairs Property=*Value*. Although the selection of properties and values may vary according to the type of media server, some properties are common to all.
  For example:
  - *Name*.
  - Any connection string may have the property (and value) *Static*=*True*.

Viz Mosart in general requires *two* connections to each video server, one from *Media Administrator* (as defined here) and one from *AV Automation* (see Video Servers, section *Working with Video Server Configuration*).

AV Automation also has a *Name* property, and the two connections to the same server should have the same, unique *Name*, This name cannot be used for any other connection.
By default, *Media Administrator* connects to *all* servers defined here.

However, if the setting **Enable dynamic configuration** is *True*, it connects only to

- The servers (identified by the *Name* property) that are part of the salvo selected in *AV Automation* (see AV Automation, **Switch** button), and
- The servers which have *Static*=*True* in the connection string.

> **Note:** When Media Administrator searches through all servers for information on a media clip, the search sequence starts with the last (highest numbered) server configured. This may impact the result of the media search if the media clip is present on more than one of the servers, and some property of the clip is not identical on these servers. In this case, the property is set from the *last* server checked (the lowest numbered server configured) from where the clip was available.

- **Video clip Server <n> Backup:** Deprecated.

Logging

- **Ignore verbose events filter** (IgnoreVerboseEventsFilter): Semicolon based list of events to ignore when using verbose logging. Default: *<none>.*
- **Log exist** (LogExist): Verifies the presence of the Viz Mosart Log Service at startup. Default: *True.*
- **Pass verbose events filter** (PassVerboseEventsFilter): Semicolon based list of events to log when using verbose logging. Default: *<none>* .
- **Use logging** (UseLogging): Passes all logging to the Log Service if true. Default: *True***.**
- **Use verbose logging** (UseVerboseLogging): Enables verbose logging if true. Default: *False***.**

Manus Admin Connection

- **Host:** Sets the hostname or IP address of the computer running the Manus Administrator application. In a typical configuration, the Manus Administrator runs on the same computer and the default value is normally

sufficient.
This value is also possible to set using the
mmhost command. Default: *localhost.*

- **Port:** Determines TCP/IP (.NET Remoting) port to establish the connection to the Manus Administrator application. Default: *8085*.

Media Router

- **Configuration filename** (ConfigurationFilename): Determines the configuration file to use for **Media Router**. Default: *MediaRouterConfigVideo.xml.*
- **Server index name**: Deprecated.
- **Use**: Uses the Media Router if true. Default: *False.*

7.1.2 Dynamic Video Server Configuration

Enabling dynamic configuration ensures that only the video servers that are currently in use by AV Automation are searched by Media Administrator.
The key setting is **Enable dynamic configuration**, described in Configuration Settings above.

Enabling Dynamic Configuration

The settings **Enable dynamic configuration** (EnableDynamicConfiguration) has several implications:

1. The *Name* properties as used in connection strings in AV Automation and Media Administrator must match. Only video servers with a name that corresponds to the video server naming in AV Automation get used.
2. The *Name* parameter is Media Administrator's method of distinguishing between the various clip metadata sources. It is also Viz Mosart's means of matching clip metadata sources and video servers.
3. Any Media Administrator connections that do not relate to a particular video server used by AV Automation. These shall be marked with Static=*True* in the connection string. A typical example is a connection to a MAM system, for clip information.

> **Note**: The *Name* property is case sensitive, this property *must* be written identically in both AV Automation and Media Administrator.
> Any other configured connections are ignored.

Example 1: Connection string to a video server (Quantel) in Media Administrator:

When dynamic configuration is enabled a corresponding video server (with name=Q1) has to be configured and used by AV Automation

```
Name=Q1;Type=Quantel;SerialNo=21234;IOR=http://quantel:@10.0.0.131/
ZoneManager.ior;Slave=10.0.0.132;Config=ClipServerQuantel.xml
```

Example 2: Static connection with a MAM system

When dynamic configuration is enabled, this connection shall always be established, regardless of the current AV Automation configuration.

```
Name=VizOne;Type=VizOne;Static=True;Server=10.0.0.136;Port=8070;Config=ClipServerJupi
ter.xml
```

7.2 Media Administrator Commands

Media Administrator, a console application, includes some command line-executable operations.

There should be no need to use these during daily operations, but they may be of help during installation or while investigating issues.

> **Info**:
> - Any changes performed with these commands are only active until Media Administrator is next restarted.
> - For persistent changes, use regular configuration, as described in Media Administrator Properties Editor and Video Server, MAM and Database Connection Strings.

The list of available commands is as follows:

- **add <clipname> <index>:** Adds a clip manually to the list of clips currently monitored. Typically used to test the installation.
- **delete <index>:** Deletes a monitored clip with the given index.
- **dumpclips <filename>:** Outputs a list of all monitored clips in an XML format. Outputs to console if no filename is given.
- **dumpprofiiler:** Shows timing information. Typically used to test the installation.
- **exit:** Exits the Media Administrator.
- **flushactionqueue:** Flushes the internal action queue of all pending actions.
- **help:** Outputs all available commands.
- **listactionqueue:** Outputs the pending list of actions to be executed.
- **listclips:** Outputs a list of all monitored clips.
- **listservers:** Outputs a list of all configured media servers.
- **mmhost <hostname>:** Sets the hostname of the Manus Administrator. Default: *localhost*.
- **postroll <roll>:** Sets/displays the current value of the post roll.
- **reconnect:** Reconnects to all configured media servers. Typically only one video server is connected.
- **reconnectadmin:** Reconnects to the Manus Administrator.
- **refresh <n>:** Refreshes the status for all monitored clips <n> times (by making a request io the server for all clips). Typically used to test the installation. Default **<n> =** `1`.
- **search <regex>:** Issues a clip search to the video server. The regex is a regular expression for clip names. Typically used to test the installation.
- **set <property>=<value>:** Sets a property value. If no property is given, a list of all supported properties is displayed. The following properties are available:
  - **NextClipAttemptDelay:** Delay in seconds when to automatically verify presence of non-existing clips.
    Note that most video servers support asynchronous notifications making this functionality unnecessary.
  - **NextServerAttemptDelay:** Delay in seconds when to try to reconnect to a video server. When the connection to a video server is not successful, this delay determines the time to wait before a new connection attempt.

    - **NextPingDelay:** Delay in seconds for pinging the connected video servers, to detect whether the servers are valid.
    - **MonitorClipInterval:** Interval for monitoring clips. A clip is polled for changes as long as any of its attributes change.
- **settings:** Opens the Media Administrator Properties Editor.
- **verbose:** Toggles verbose output. In Verbose mode, more information is logged to the console.
- **version:** Outputs Media Administrator version and some system information.

8 Overlay Graphics Interface

- Overlay Graphics Interface Overview
- Overlay Graphics Configuration and Use

The Overlay Graphics Interface provides control and monitoring of overlay graphics for all Viz Mosart-approved graphic engines.

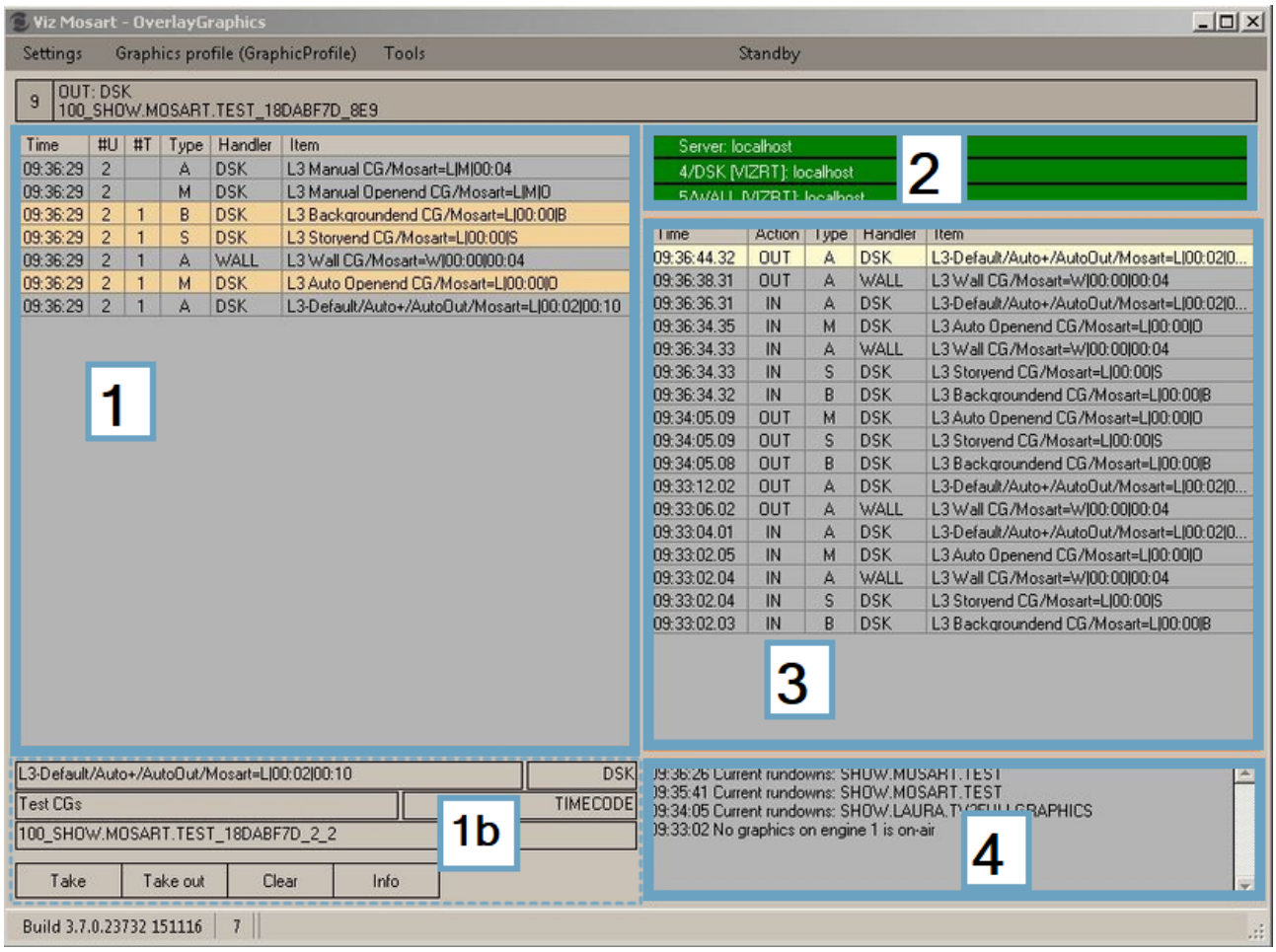
- For a list of supported graphics devices, see Overlay Graphics Types.
- Configuration is done in the Overlay Graphics Configuration Properties menu.

Start **Mosart Graphics Interface** by double-clicking the application icon:

or by navigating to the installation directory (by default `C:\Program Files (x86)\Mosart Medialab\Mosart Server`) executable `MMOverLayGraphicsInterface.exe.`

> **Info**: The Overlay Graphics interface is *only* used for overlay graphics. *Full-frame* graphics are controlled through the AV Automation application.

A menu bar at the top offers **Settings**, **Graphics profile**, **Tools** and **Standby**.

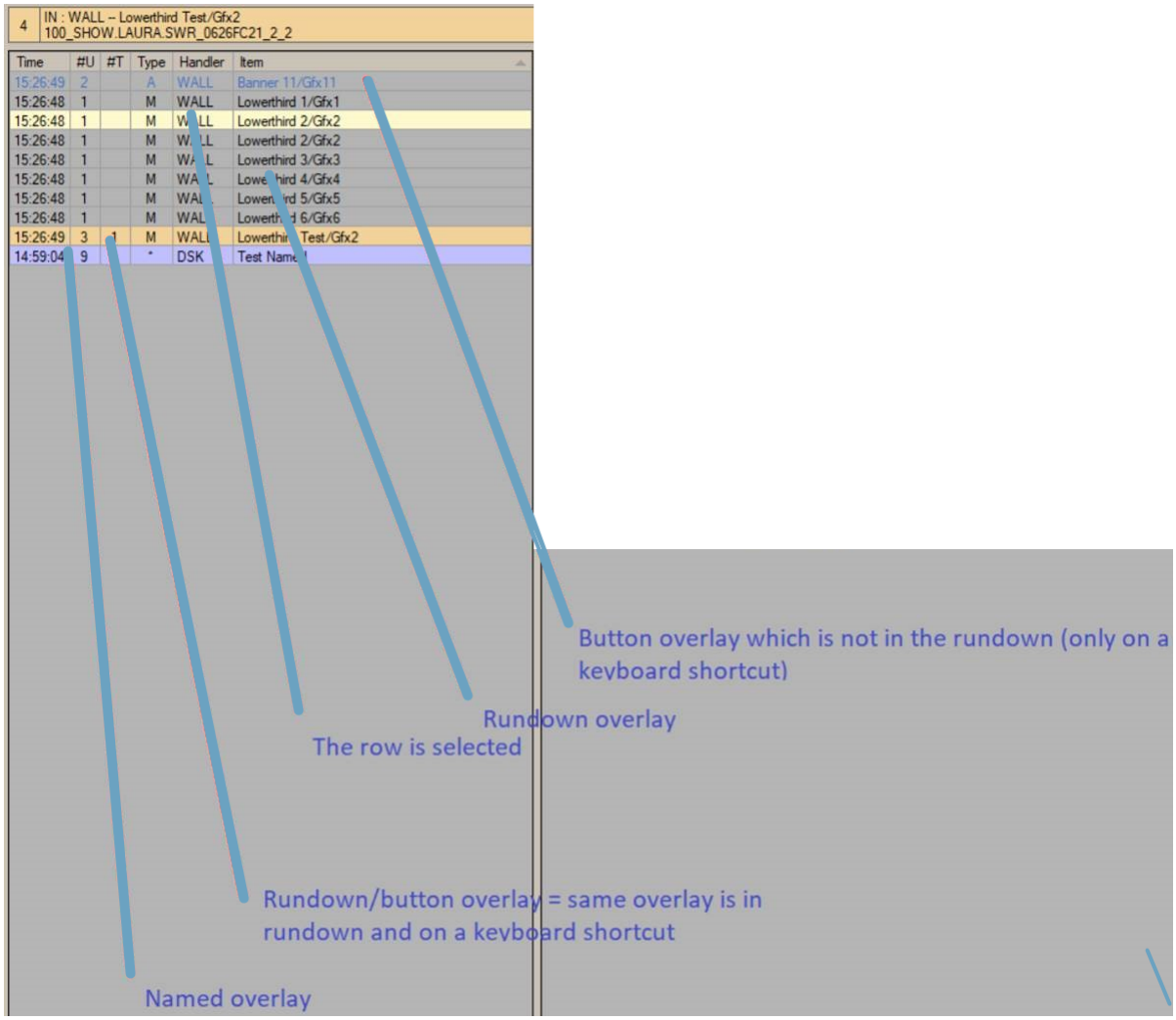


8.1 Overlay Graphics Interface Overview

The Overlay Graphics Interface GUI consists of four main areas (**[1]** to **[4]** in the illustration).

A status bar at the foot of the UI provides information on product version number and the total number of graphics in Viz Mosart (rundown overlays + named overlays + button overlays. *7* in the example above).

8.1.1 [1] List of Overlay Graphics in the Rundown



The main column labels are:

- **Time:** The time when the graphic was added in Overlay Graphics (usually at rundown import).
- **#U:** The number of times this graphic has been updated.
- **#T:** The number of times this graphic has been taken.
- **Type:** The graphic type. Can be:

Type	Description
A	Auto Out (a Manual graphic can also be Auto Out)
B	Background End
S	Story End
M	Manual (Open End)
L	Last Out
-	Stay Always
>	If template is **STORYSTART**
***	None of the above

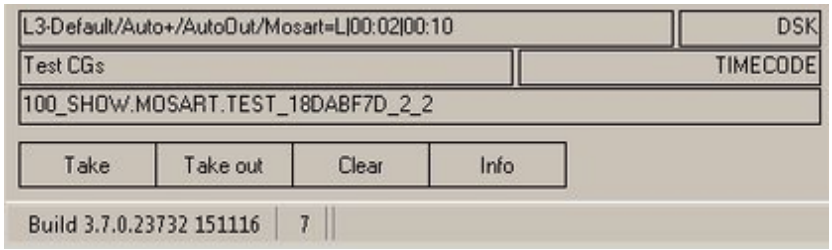
If the graphic is a *locator*, then the type is suffixed with an "**L**".

> **Info:** *Locator* is a special graphics item that is *linked* to a crosspoint. When taking the crosspoint from a Viz Mosart template, either as a switcher crosspoint or keyed crosspoint (in a DVE box), the graphics are also (re-)taken.

[1b] Details of a Selected Graphic from the List of Overlay Graphics

Selecting an overlay graphic from the list displays further details.

> Note that the newest row is selected automatically whenever a graphic is taken, so a manual selection may be replaced by a later take.



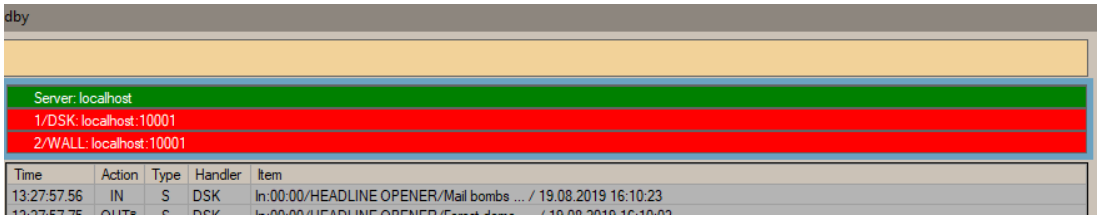
Explanation of Displayed Elements

Element	Description
	Graphic title

Element	Description
	Handler name (the destination or the channel for this graphic)
	Graphic description
	Graphic ID **Info**: This is the *Viz Mosart* Graphic ID, not the graphic ID from graphic database (Graphic Hub/Oracle)
	Action Buttons **Take:** Takes graphic on air **Take out:** Takes graphic off air **Clear:** Takes out all taken graphics **Info:** Opens a window where the graphic MOS item can be inspected, and potentially edited and updated. The editing and updating of graphics in this window is a legacy feature that is not recommended to use. Be aware that not all property changes in this view are effectuated, and if not, they are silently discarded. For a **Named Overlay** that uses placeholders (for example `{story:slug}`), Viz Mosart re-resolves the content on every take. All takes of the *same* Named Overlay share one Viz Mosart Graphic ID, so only the most recent version is kept: **Info** shows that overlay's latest content, not the content as it was when an earlier row was logged. Selecting a *different* Named Overlay shows that overlay's own content correctly, and a Named Overlay without placeholders is static and always shown correctly.

8.1.2 [2] Server Status

Viz Mosart server and configured graphic engines statuses are displayed (*Green*: OK, *Red*: Failure) in the panel at the top right:



8.1.3 [3] General Log

The center-right contains a log of all actions performed on the overlay graphics either triggered by Viz Mosart or by clicking **Overlay Graphics** buttons.

To clear the General log area

- Double-click inside it.

8.1.4 [4] Event Log

The bottom right contains an event log with messages, warnings and errors from the Overlay Graphics application.

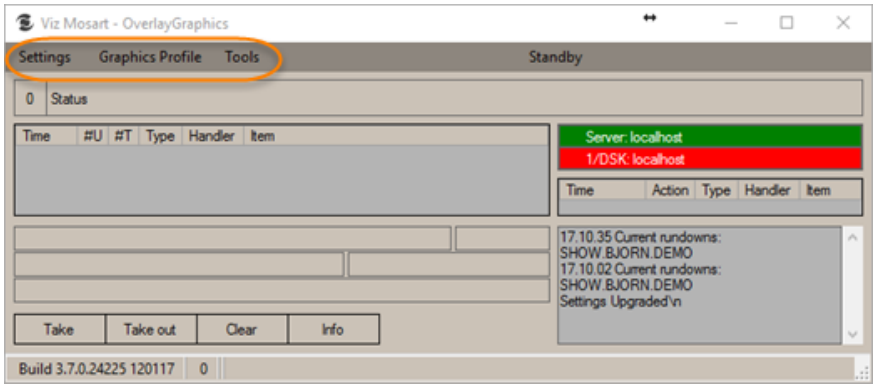
To clear the Event log area

- Double-click inside it.

8.1.5 System Information

The information line at the bottom displays application version, number of graphics in the table and whether the Mosart Media Router (MMR) is in use:

8.1.6 Menu Bar



Settings

Properties: Opens Overlay Graphics Configuration. See Overlay Graphics Configuration and Use

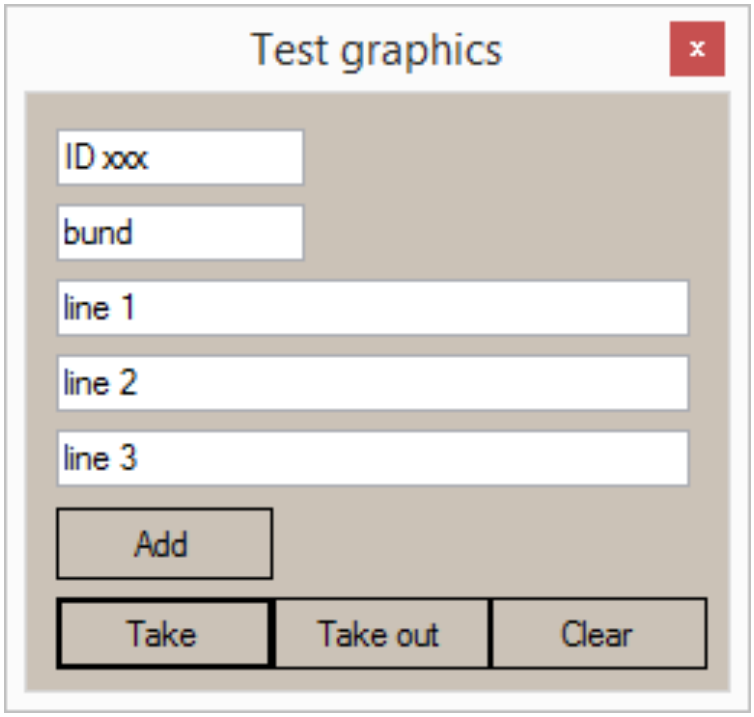
Graphics Profile

Select the required graphics profile.
See Graphics Profiles Tab.

Tools

When things don't go as expected, there is a debugging option.

Editor: Opens a **Test graphics** window.



Instrumentation Panel: Displays all device commands in real time, as sent to the various connected devices.
See Mosart Instrumentation Panel.

Standby

Toggles the graphics engines in or out of standby.

To take any controller into standby

- From the menu bar, click **Standby**.
  This is a toggle button. All engines linked to the controller are set/unset to standby by clicking **Standby**.

> **Info**:
> - Graphic controller standby actions can also be performed from the Viz Mosart UI.
> - You can take any engine *individually* into standby (only in the OGI application) while the other engines in the same controller are active.

8.2 Overlay Graphics Configuration and Use

See the respective sections:

- Mosart Graphics Destinations and Takeout Logic
- Sending Lower Third Graphics Details to an NRCS
- Overlay Graphics Configuration Property Tabs
- Overlay Graphics Types
- Named Overlay Graphics
- Managing Overlay Graphics Controllers and Engines

Copyright © 2026 Vizrt

8.3 Mosart Graphics Destinations and Takeout Logic

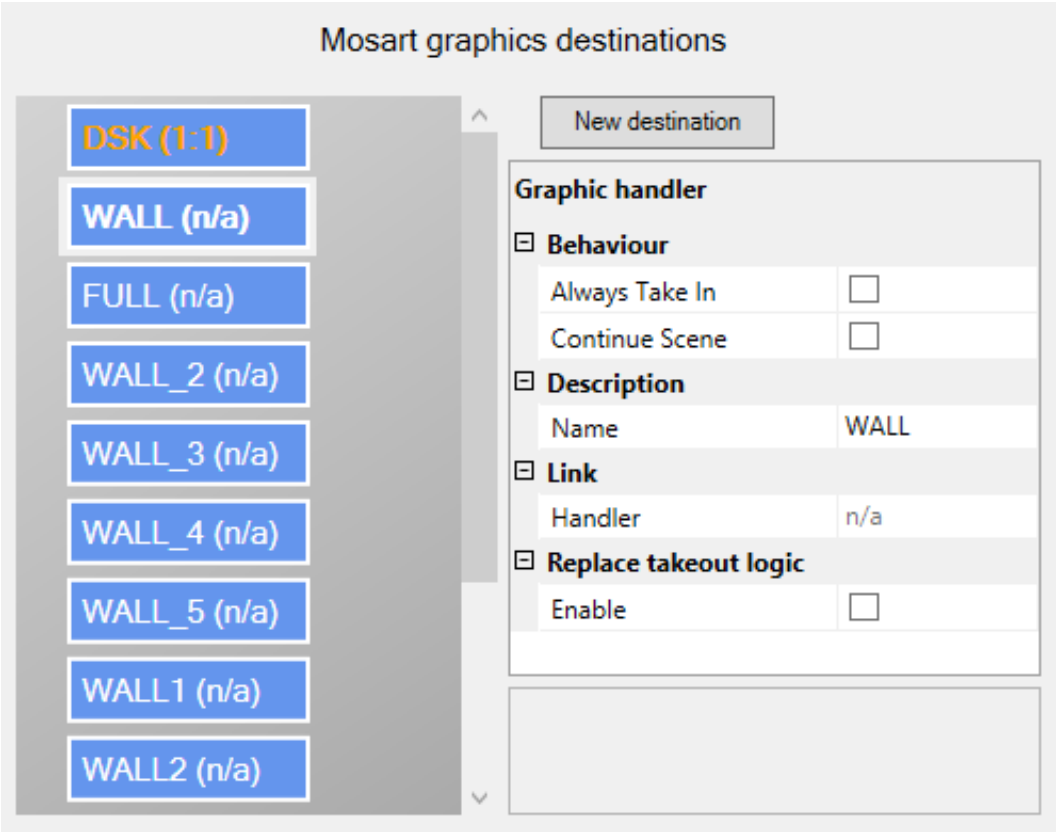
This section presents configuration of overlay graphics for various system setups and the use of takeout logic.

- Mosart Graphics Destinations
  - Overlay Graphics Interface
  - Graphic Handler Properties
  - Mirrored Graphics
- Takeout Logic
  - Introduction
  - Enabling Take Out Logic
  - Prerequisites for Take Out Logic

8.3.1 Mosart Graphics Destinations

Overlay Graphics Interface

The Overlay Graphics Interface provides definitions to a set of standard Mosart graphics destinations. These include WALL handlers (WALL, WALL_2 to WALL_5), FULL, and TABLE. It is recommended that you keep each definition as-is.



To Add a New Mosart Graphics Destination

- Click **New destination**, then fill in the **Graphic handler** properties.

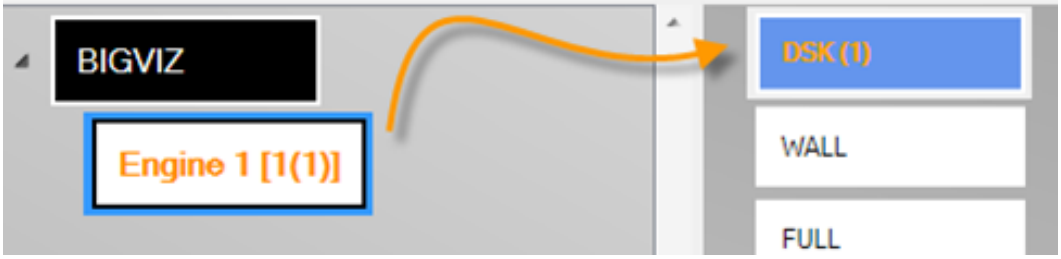
Graphic Handler Properties

The properties displayed depend on the graphic handler used. For some handlers (e.g. Viz Engine) the full set of or possible properties will be shown. For others, a simplified set of properties will show: **Description** and **Link**.

- **Behavior**:
  - **Always Take In**: Turns off the logic that does not take in a graphic element that is already taken and no other graphic element has been taken since.
  - **Continue Scene**: Use continue for take out of scene based graphics.
- **Configuration**:
  - **Preview Engine**: Handler name used for preview.
- **Description**:
  - **Name**: Name of the graphics destination.
- **Link**:
  - **Handler**: Shows the engine that is linked to this destination. The engine is shown as Handler ID: Handler Name.
- **Replace takeout logic**:
  - **Enable**: Reads out the Transition Logic context of elements and use this to either suppress take out or re-take items. Take outs will be suppressed if another item with the same context was taken before the scheduled for take out action. Take outs will be replaced with re-takes if the item scheduled for take out replaced an item with the same context.

To Link an Engine to a Destination

- To create a link from an engine to a destination, drag an engine node to the preferred destination.
  The destination label and Handler property will be updated to show the Handler ID and Name of the engine.



To Remove Engine Links from a Destination

1. Mark an engine.
2. **Right-click** to display the context menu context menu and select **Remove**.

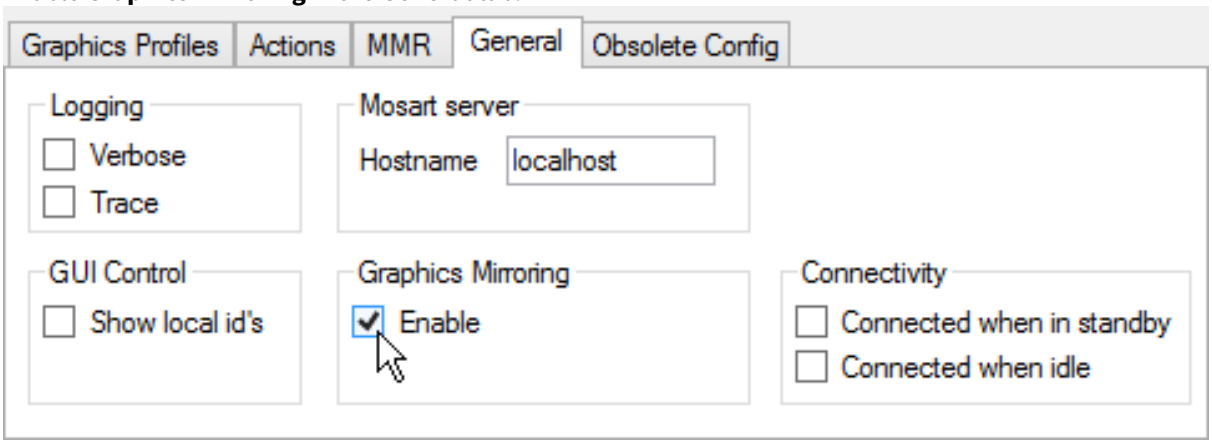
Mirrored Graphics

Mirrored Graphics Playout

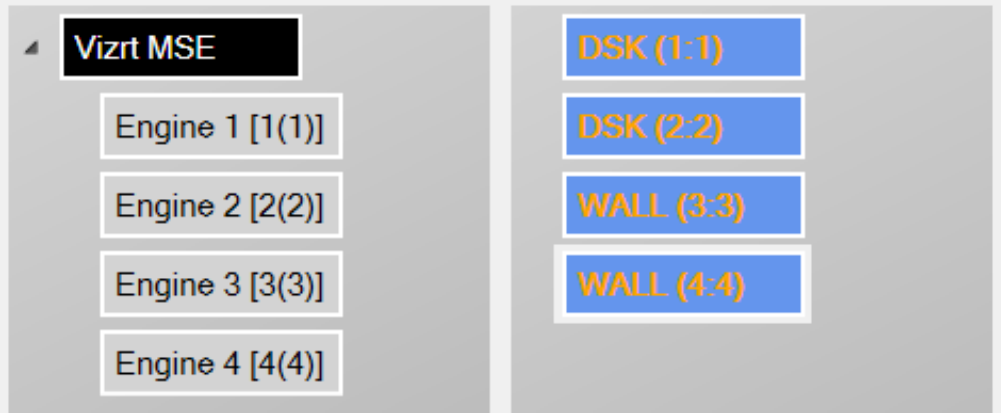
The feature of playing out the same graphics through multiple graphics engines is called Mirrored Graphics Playout. This functionality is supported both for overlay graphics and full-screen graphics (AV Automation). Here we describe the approach used in Overlay Graphics Interface.

> **Info**: For using Mosart Media Router (MMR) in conjunction with Mirrored Graphics, see Mosart Media Router (MMR) and Mirrored Graphics below.

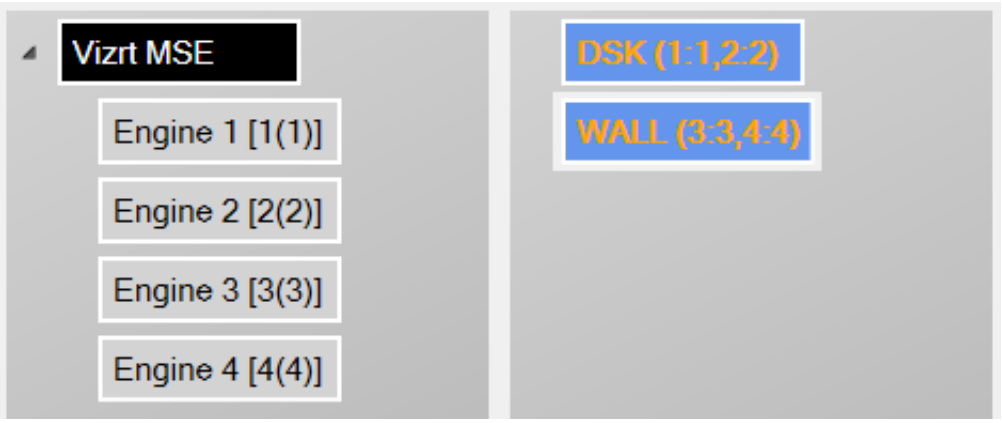
1. Enable **Graphics Mirroring** in the **General** tab.



2. Set up multiple graphics engines with the same destination.
   - **Example #1**: To configure mirrored graphics for a destination, create two destinations with the same name and assign a different engine to each destination.
     In the example below, mirrored graphics is enabled for **DSK** and **WALL** destinations.



   - **Example #2**: To configure mirrored graphics for a destination, drag-and-drop two different engines to the destination.
     In the example below, mirrored graphics is enabled for **DSK** and **WALL** destinations.



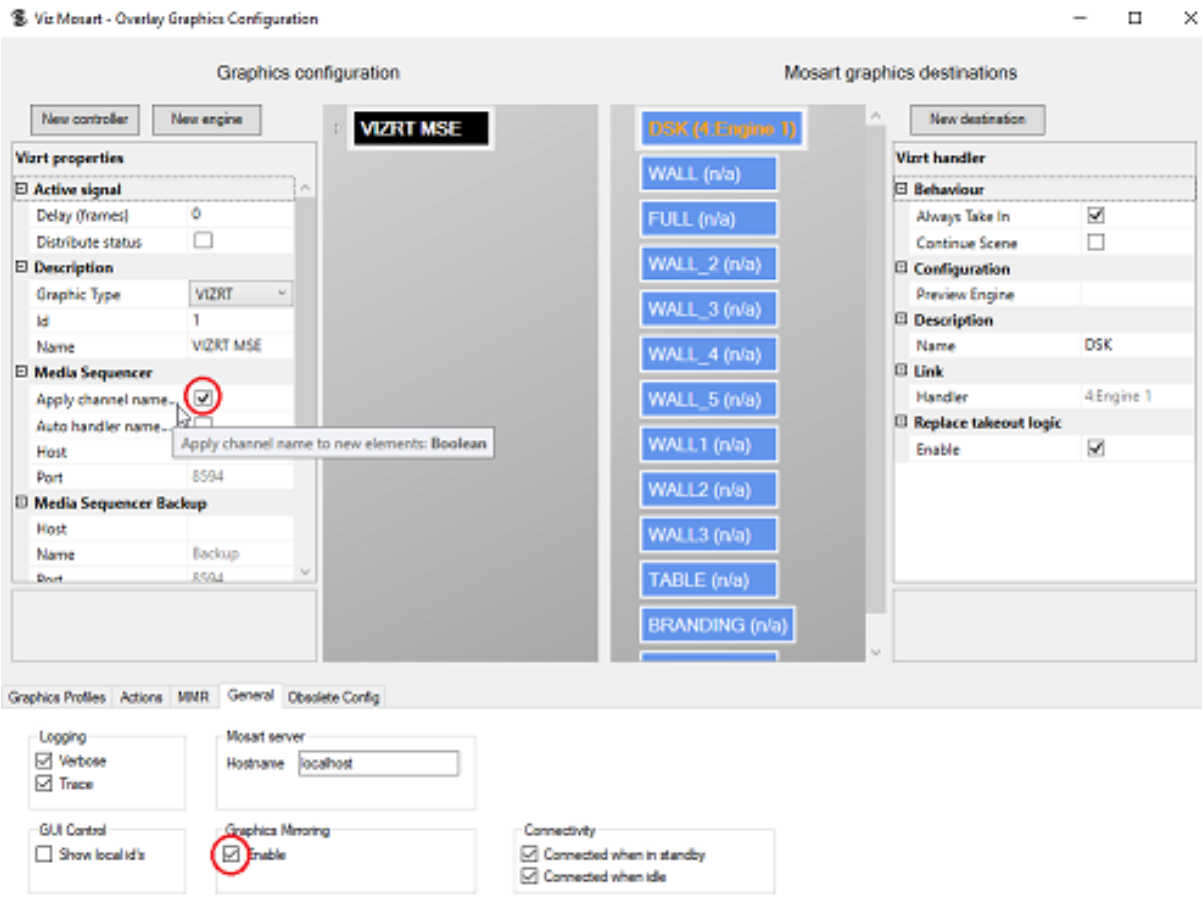
3. (Optional): Configure actions to be triggered when changing the active mirrored graphics engine. Do this from the **Actions** tab.

Mosart Media Router (MMR) and Mirrored Graphics

If your setup employs MMR for mirrored graphics, the integration requires additional configuration to ensure that all new graphics elements are tagged with their corresponding destinations in the Mosart playlists.

- Ensure the **ApplyChannelNameToNewElements** property located in `DefaultGraphicsConfiguration.xml`, is set.

This is also configurable from the UI:



This is described in detail in the *Viz Mosart Media Router Guide*, in the section **MMR Configuration of Vizrt Graphics**.

8.3.2 Takeout Logic

- Mosart Graphics Destinations
- Takeout Logic

Introduction

For overlay graphics, Viz Mosart supports four different methods to take out any graphics that are currently on air:

1. **Automatic**: Timed, where graphics are taken out automatically after a fixed duration, provided in a timecode.
2. **Background End**: The graphic is taken out automatically, when the next story item is taken.
3. **Story End**: The graphic is taken out automatically when the first story item of the next story is taken
4. **Manual**: Open ended, the graphic is taken out manually. Also called a *non-auto out*.

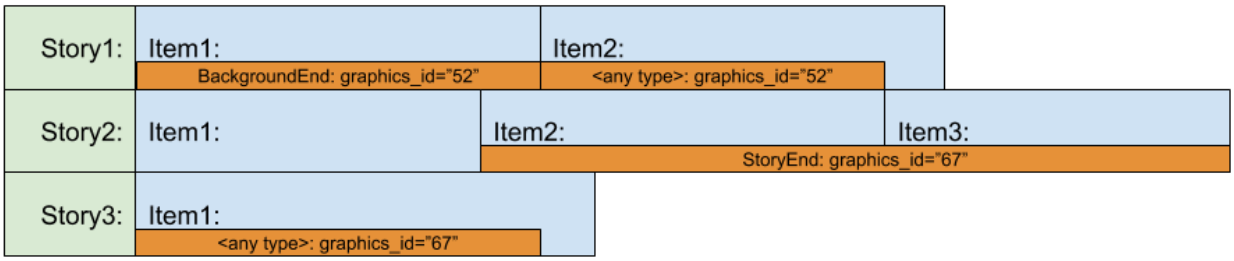
For the *Background end* and *Story end* cases, there is a potential issue if the graphic to be taken out reappears in the next story item. When taking the next story item, the standard behavior of such graphics is:

1. Take out the graphics due to the end of scope (either *Background end* or *Story end*)

2. (Re)take in the graphics since the graphics is also present in the story item to be taken.

This leads to a visual artifact on air, since the graphics will be take out and retaken within a short time interval.

This is illustrated in the following figure:



Example 1: Background End

Story1 consists of two story items, *Item1* and *Item2*.

- In *Item1* there is an overlay graphics of type *BackgroundEnd.*
- In *Item2* the same graphics is reused. The type is not relevant here.
- When the user takes *Item2* on air, the normal behavior is to take out the graphics (out of *Item1*) and then (re)take it (also part of *Item2*).
- If Take Out Logic is enabled, the graphics will remain on air during the transition from *Item1* to *Item2.*

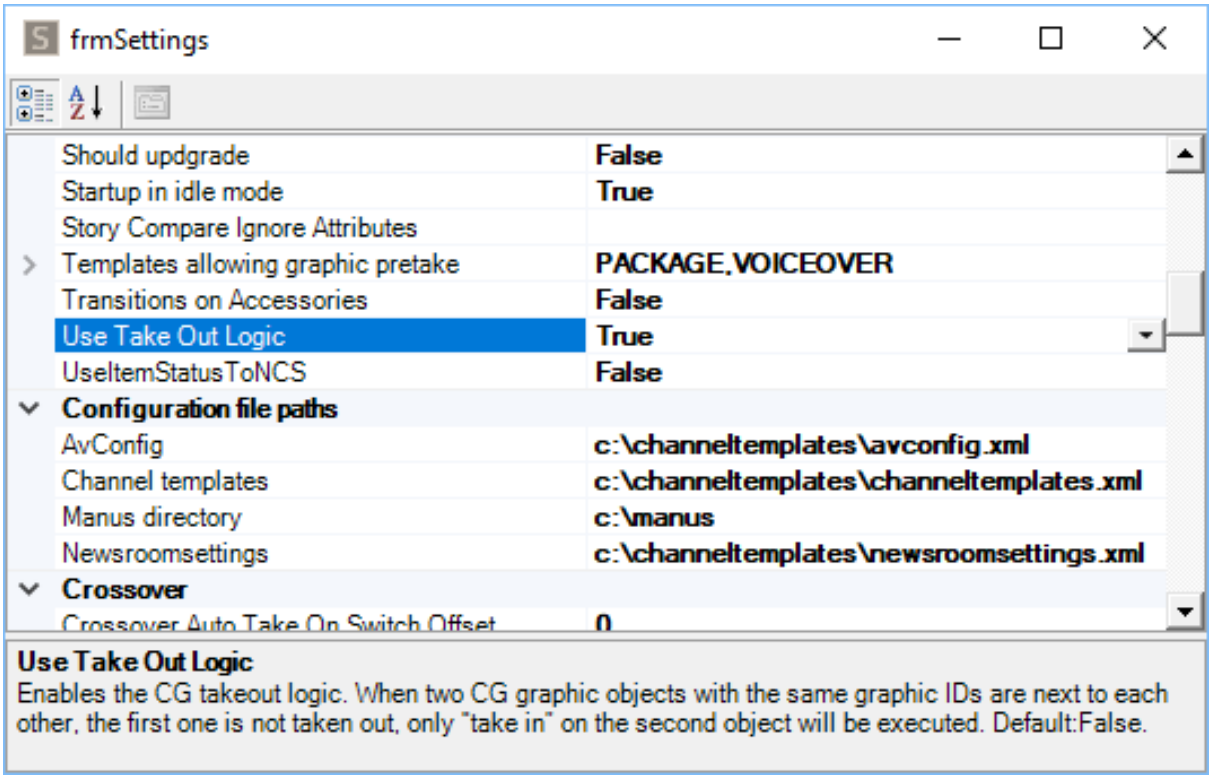
Example 2: Story End

Story2 consists of three story items, *Item1*, *Item2* and *Item3* and the next story, *Story3* consists of one story item, *Item1*

- In *Story2.Item2* there is an overlay graphic of type *StoryEnd*. I.e. this graphic is scheduled to be taken out at the end of the story.
- In *Story3.Item1* the same graphics is reused. The type is not relevant here.
- When the user takes *Story3* on air, normal behavior is to take out the graphics (out of *Story2.Item3*) and then (re)take it (also part of *Story3.Item1*).
- If take out logic is enabled the graphics will stay on air during the transition from *Story2* to *Story3.*

Enabling Take Out Logic

- Take out logic is enabled via setting **Use Take Out Logic** in Manus Administrator as shown in the figure below:



Info: Enabling take out logic applies to *non-Vizrt graphics* only.

Prerequisites for Take Out Logic

For take out logic to work, there must be a unique way of identifying the graphics.

To identify graphics for takeout logic

There are two ways to identify graphics for takeout logic.

1. Use the *graphics_id* graphics property. The *graphics_id* property is used by most graphics drivers to do actions on the graphics. Therefore it can normally be used as a unique identifier for the graphics. For most graphics systems used by Viz Mosart, there is a defined way of setting this property when receiving graphics from NRCS.
2. Use the *graphics_category* graphics field property. This is a special property for the take out logic. Use this if graphics with different *graphics_id* shall be treated as the same graphics and handled as part of the take out logic. This normally requires custom field mapping in configuration file *newsroomsettings.xml*.

Copyright © 2026 Vizrt

8.4 Sending Lower Third Graphics Details to an NRCS

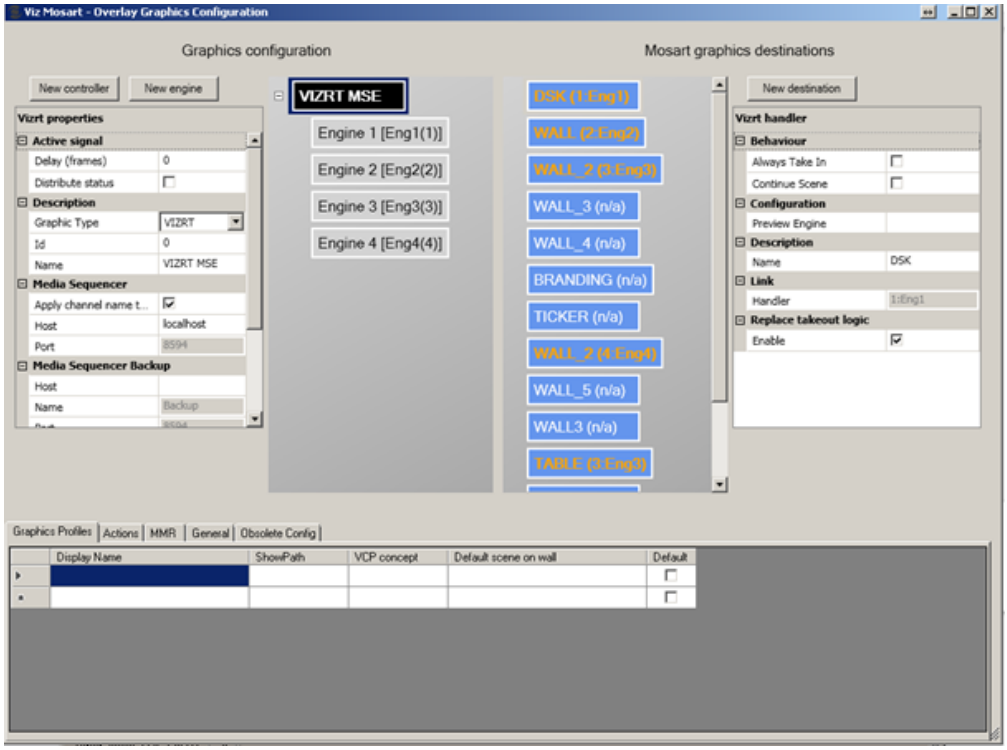
You can send details to a MOS-based NRCS as MOS objects.

For all template details as MOS objects in the NRCS, see section Mosart Templates Feedback to the NRCS.

8.4.1 Managing Destinations

Active Destinations

The list of channels contains the active destinations is configured in the Overlay Graphics Interface (OGI). *Active destinations* are destinations linked to a graphic engine.



In the configurations above, the following engine-destination mapping is performed:

Destination	Engine number
DSK	1
WALL	2
WALL_2	3

Destination	Engine number
WALL_2	4
TABLE	3

MOS Format

The list of channels in the LOWERTHIRDS MOS object will be:

```
<keys name="channels">
<k>WALL_2</k>
<k>WALL</k>
<k>TABLE</k>
<k>DSK</k>
</keys>
```

The active channels from Overlay Graphics are sent to the NRCS only if the Overlay Graphics Interface is started. This is the recommended workflow to correctly send the channels configured in Overlay Graphics Interface to NRCS.

To send configured OGI channels to the NRCS

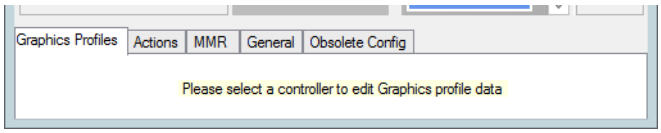
The steps in square brackets are optional.

1. Start Manus Administrator
2. Start Overlay Graphics Interface (OGI)
3. [Perform any required configuration changes in **OGI > Settings**]
4. [Restart OGI or do **Ctrl+Shift+G** if configurations are changed]
5. [Verify that %localappdata%\Temp\Mosart Medialab\MOS State\ MosTemplateCache.xml contains expected data]
6. Execute mosReqObjList or mosReqAll from the NRCS

8.4.2 Detailed Description of MOS Objects

See the Appendix for code-level descriptions of Structure of the MOS object with Lowerthird Information.

8.5 Overlay Graphics Configuration Property Tabs



The tabs functions are:

8.5.1 Working with Overlay Graphics Configurations

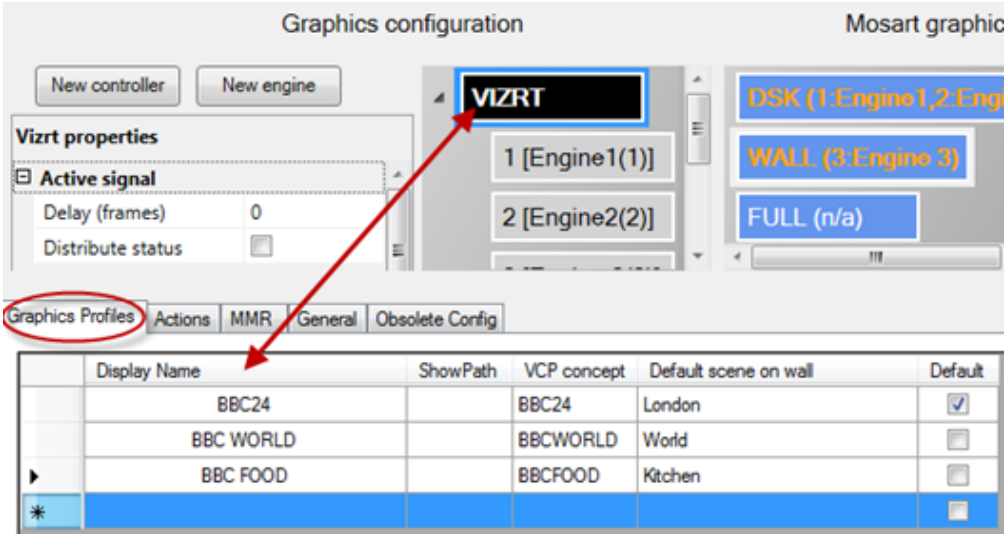
- To change a Graphics profile
- To set up event-action rules
- To configure the Mosart Media Router
- To configure general settings
- To convert an incompatible overlay graphics configuration

Graphics Profiles Tab

The selection of graphics profile affects the look of all graphics used for the specific show. Viz Mosart handles which concept the graphics devices use (on Vizrt graphics systems this technique is called **concept override**, on other graphics systems Viz Mosart uses its own concept override technique).

To change a Graphics profile

1. Navigate to **Settings > Properties > Graphics Profiles** tab.
2. Select the controller (black) that you want to configure.
   The **Graphics Profiles** list changes dynamically, depending on which controller is selected.
3. When the graphics profile is changed in the Overlay Graphics Interface (or the Viz Mosart GUI), all the controllers are scanned.
   If a controller has the selected graphics profile, the show path, VCP concept (Viz only) and default scene for wall values, are all changed.
4. Remember to restart the Overlay Graphics Interface application for any changes to take effect.



- **Display Name:** Name of the Graphics Profile that is displayed to the user. The Graphics Profiles are displayed as menu options both in the Mosart GUI and in Overlay Graphics.
- **ShowPath:** Path to the files the graphics engine uses. Absolute path where graphics are located within the graphics engines. Dependent upon graphics type.
- **VCP concept:** Name of the Viz Pilot concept that should be used when working with the selected profile (Vizrt only).
- **Default scene on wall** Scene name to use as default for wall graphics. Dependent upon Graphics Type.
- **Default:** Selects the graphics profile to use as default. The default graphics profile is used initially before any profile is set from the user.

Click on the Graphics Profile tab to switch between profiles. The default profile is displayed in bold, and the active profile is marked with a check mark, and is displayed in parenthesis on the menu itself.

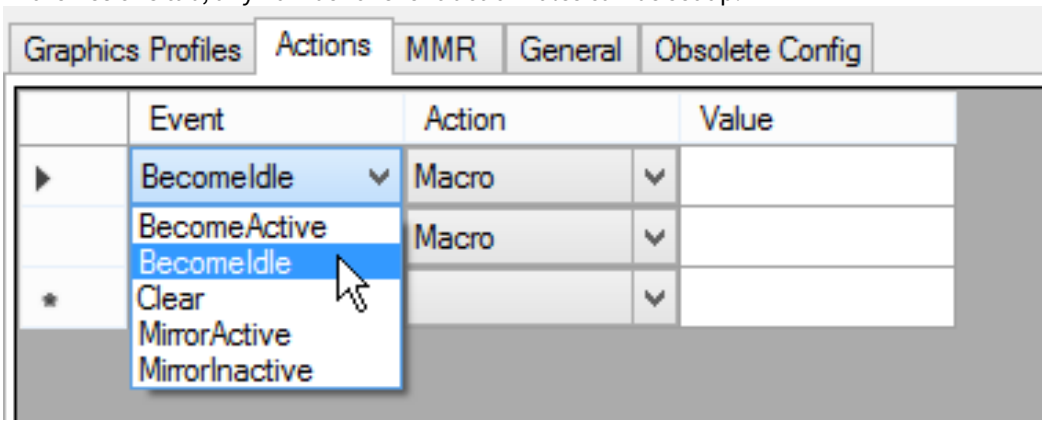
-

Actions Tab

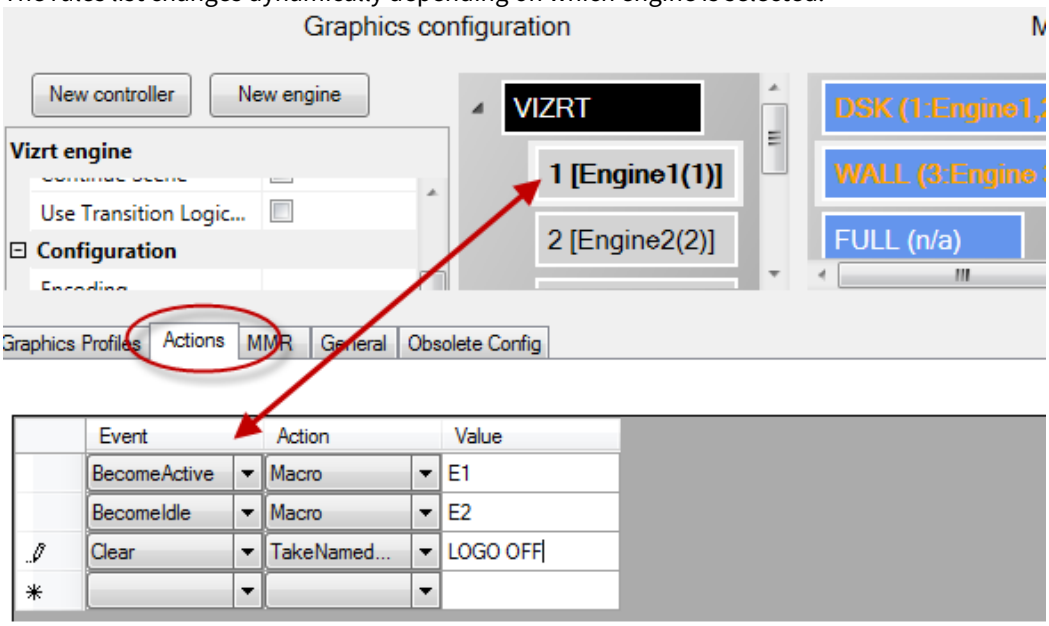
Configurations on this tab control *Event* and *Action* rules.

To set up event-action rules

1. Navigate to **Settings > Properties > Actions** tab.
   In the **Actions** tab, any number of event-action rules can be set up.



2. Select the engine that you want to configure these actions on.
   The rules list changes dynamically depending on which engine is selected.



3. Set required event-action rule by selecting parameters.
   You must define three parameters to set up an event-action rule:

- **Event:** Specifies the event that will trigger an action. The following events are available:
  - **BecomeActive:** Triggers an **Action** whenever the Overlay Graphics Interface state changes from **Idle** to **Active**.
  - **BecomeIdle:** Triggers an **Action** whenever the Overlay Graphics Interface state changes from **Active** to **Idle**.
  - **Clear:** Triggers and **Action** whenever a **Clear** event is received to clear all overlay graphics. A clear event is typically received when a rundown is reloaded or deleted.

  - **MirrorActive:** Triggers an **Action** when the mirroring graphics engine becomes active (for mirroring). Can only be used only when using Mirrored Graphics Playout.
  - **MirrorInactive:** Triggers an **Action** when the mirroring graphics engine becomes inactive (for mirroring). Can only be used only when using Mirrored Graphics Playout.
- **Action:** Specifies which action to trigger for the given event. Two actions are available:
  - **Macro:** Invokes a macro on the given engine. The macro to take is specified in the **Value** parameter.
  - **TakeNamedOverlay:** Invokes a named (i.e. constant) overlay graphic on the given engine. The named overlay graphic to take is specified in the **Value** parameter.
- **Value:** Optional value depending on selected **Action**.

As shown above, there may be multiple event-action rules for a controller, for example:

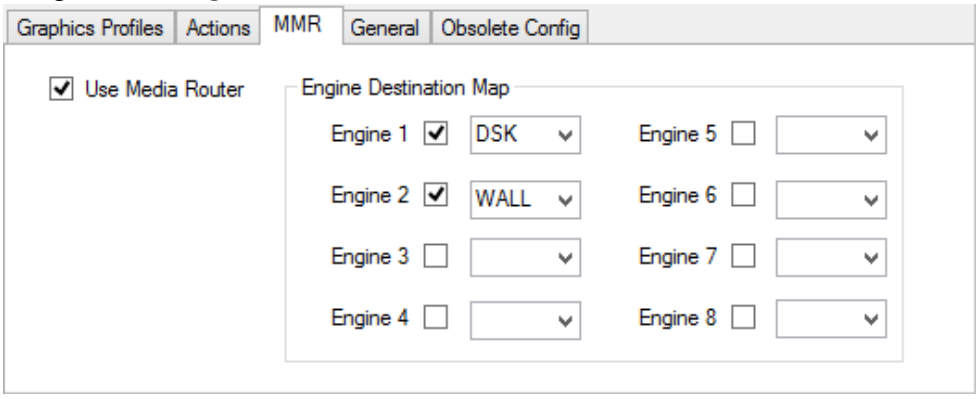
-
  - **Add** a new event-action rule by setting values in the row at the bottom, marked with a star '*****'.
  - **Selecting** the corresponding row is done by clicking in the column left of the **Event** column.
  - **Delete** existing event-action rules by selecting the corresponding row, and then press **DELETE** on the keyboard.

MMR Tab

Configurations on this tab affect behavior of the Mosart Media Router.

To configure the Mosart Media Router

- Navigate to **Settings > Properties > MMR** tab.



Explanation of configurable fields

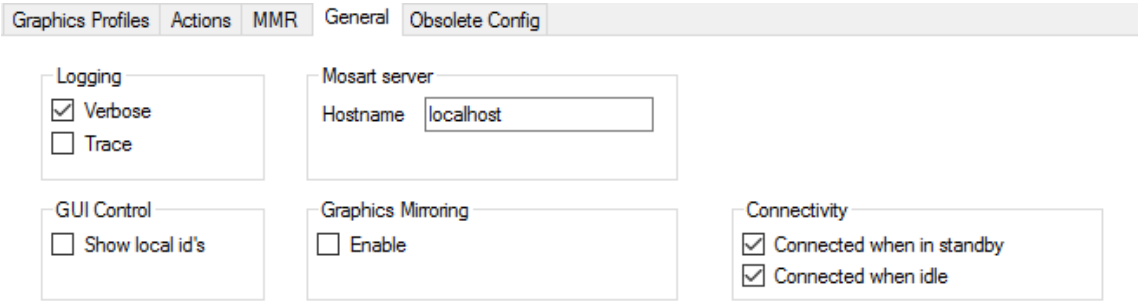
- **Use Media Router:** To use the Media Router (MMR), this must be enabled (checked). The use of Media Router by Overlay Graphics Interface must also be enabled in General **> Media Router**.
- **Engine Destination Map:** Maps between Engine number and graphics destination. The Engine number is part of the Media Router configuration and is given as a *DeviceName* property.
  For more information see the *Viz Mosart Media Router Guide*. Use the check boxes to select a maximum of eight engines. In the example above the **MMR** configuration specifies *two* overlay graphics engines with numbers *1* and *2* respectively.

General Tab

On this tab are several configurations that affect Viz Mosart Overlay Graphics operation and behavior.

To configure general settings

- Navigate to **Settings > Properties > General** tab.



Explanation of configurable fields

- **Logging**
  - **Verbose:** Increases the detail level of information sent to the Viz Mosart log if enabled.
  - **Trace:** Activates internal tracing (more details are sent to the debug view) if enabled. Used for debugging purposes only.
- **GUI Control**
  - **Show local id's:** Enables display of graphics ID in the OverlayGraphics GUI. Shown in dedicated columns in both repository and log tables.
- **Mosart server**
  - **Hostname:** Defines the hostname or IP number of the Manus Administrator. Default: *localhost*.
- **Graphics mirroring**
  - **Enable:** Allows simultaneous control of two different graphics engines driven in mirroring by Viz Mosart. When enabled, both engines will receive "take" commands at the same time. For more details, see Mirrored Graphics Playout
- **Connectivity**
  - **Connected when in standby:** Maintains connection to graphics devices when in standby or idle modes if enabled.
  - **Connected when idle:** Maintains connection to graphics devices when put on idle, for example when switching from main to backup if enabled (Default = *disabled*).

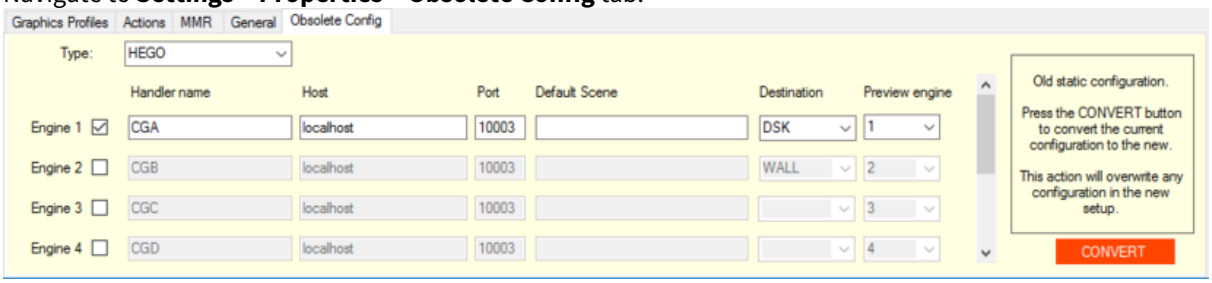
> **Note:** A similar parameter is available for full screen graphics, see **_GraphicsConnectedWhenIdle _** in AV Automation Settings (see Graphics).

Obsolete Config Tab

This tab is used to convert overlay graphics configuration from Viz Mosart 3.4 or earlier versions.

To convert an incompatible overlay graphics configuration

1. Navigate to **Settings > Properties > Obsolete Config** tab.



> **IMPORTANT!** If you have configuration based on Viz Mosart 3.5 or later, then *DO NOT* use the **Obsolete Config** tab.
> Only if you have a configuration based on Viz Mosart *3.4 or earlier*, is it strongly recommended to convert your configuration to the new standard.

   This tab displays the configuration parameters as they were in Viz Mosart 3.4 and earlier, and is used to update old configurations to the new standard.
2. Click the **CONVERT** button.
   This overwrites any existing standard configuration.

The fields in this tab are:

- **Type:** The type of graphics used
- **Engine 1-8 (check box):** Check the graphics engines to be active. When converting to new configuration, only active engines are converted.
- **Handler Name:** Default CGA, CGB etc.
- **Host:** Hostname or IP-address of graphics engine.
- **Port:** TCP/IP port number of graphics engine.
- **Default scene:** Specifies the graphics destination to be associated with the graphics engine. The graphics destination is assigned to graphics received from NCS.
- **Preview engine:** The number of the engine to use as a preview engine.

Copyright © 2026 Vizrt

8.6 Overlay Graphics Types

Each graphic provider is stored as a **Graphics Type.** Several Graphics Types are currently supported. These include:

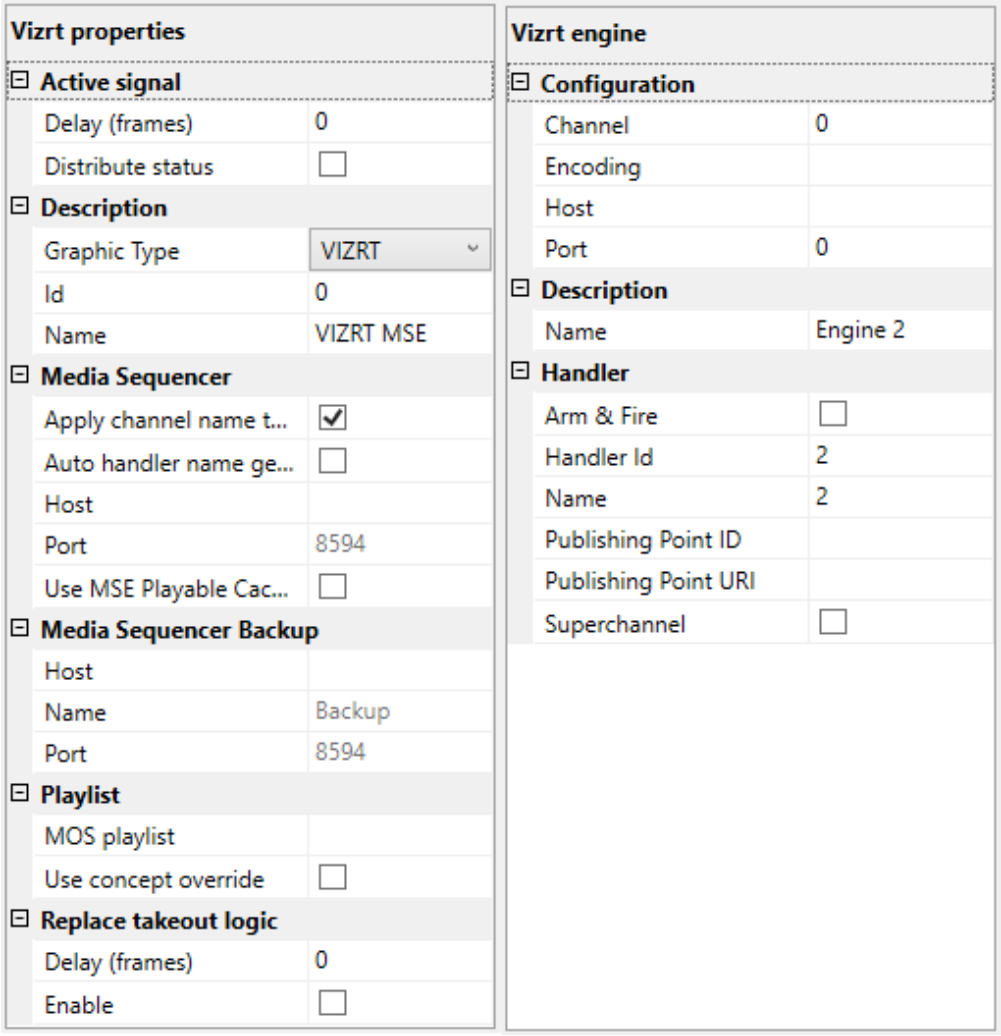
- VIZRT
- VIZ FLOWICS
- CHYRON
- DEKO
- ORAD
- PIXEL POWER CC
- XPRESSION
- BIGVIZ

All operations are managed from the **Overlay Graphics Configuration** menu.

- Open the **Overlay Graphics Configuration** menu by opening the console application **Viz Mosart Overlay Graphics Interface > Settings > Properties**.

8.6.1 VIZRT

Graphic Type = Vizrt



Vizrt graphics are provided through a Media Sequencer (MSE).

- **Vizrt properties:** Properties in addition to Common Graphic Controller Properties:
  - **Media Sequencer**
    - **Apply channel name to new elements:** Assigns a channel name automatically to new graphics elements.
    - **Auto handler name generation**: If selected, Mosart creates the handler names in Media Sequencer (MSE) using the template stored in setting *OvXHandlerName* from file the *GraphicsDefaultConfiguration.xml* located in the installation folder (usually at C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles). Default: *false*.
      The handler names are mainly used in the Vizrt graphics workflows to identify a Viz Engine with MSE properties.
    - **Host:** Hostname or IP address to the Media Sequencer.
    - **Port:** IP-port to the Media Sequencer. Normally `8594` (Media Sequencer TreeTalk protocol) or `8580` (REST protocol).

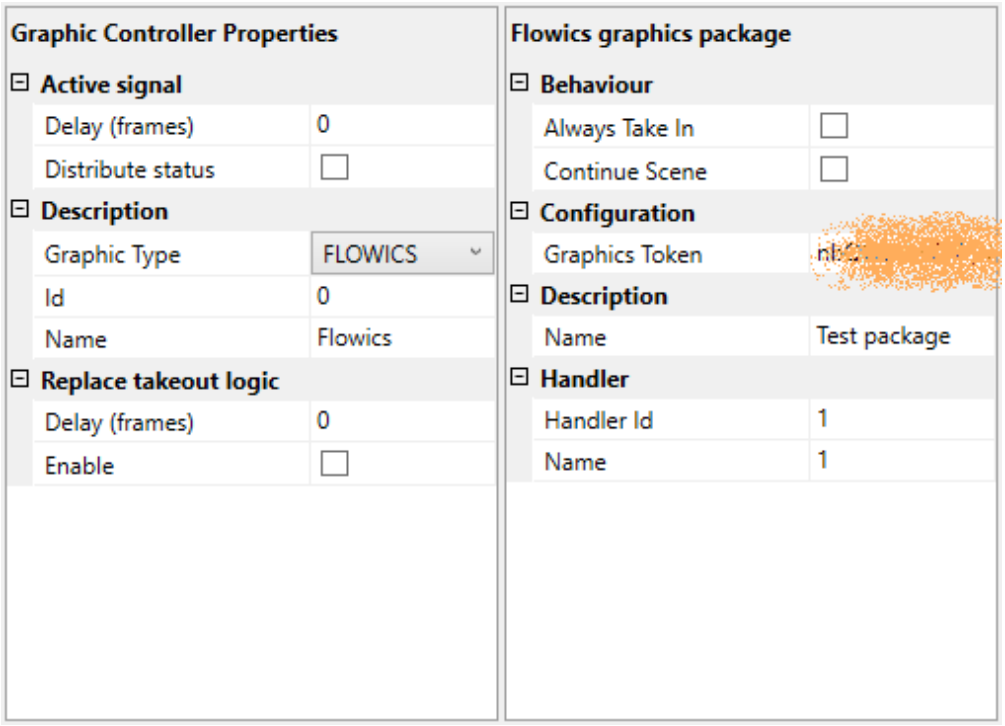
        - **Use MSE Playable Cache**: When enabled, the *playable_cache* plugin in the Media Sequencer (MSE) will be used. For further details on how the setting should be used, please contact Vizrt Support. This setting requires MSE *5.4 or higher.*
    - **Media Sequencer Backup**
        - **Host/Name/Port**: For an optional backup Media Sequencer (MSE). Same properties as for MSE.
          Note that property **Apply channel name to new elements** (see above) is common to *both* main and backup Media Sequencers.
    - **Playlist**
        - **MOS playlist:** Name of playlist in MSE where graphics elements are stored.
        - **Use concept override:** If concept override is enabled, the currently selected graphics concept will override any concept defined within the graphics element. Use the **Graphics Profile** tab to define the various graphics concepts.
- **Vizrt engine**
    - **Configuration**
        - **Host:** Hostname or IP address of the **Vizrt Engine**. The input can be specified as just hostname, just IP Address or a combination of hostname:Port or IP address:Port.
          If no port value is specified as part of the **Host** field, the port from the **Port** field will be used. However, if no port value has been specified in either of the **Host** or **Port** fields, the default port value will be *6100*.

          > **Info:** If a port value is specified in the **Host** field, this value will override whatever port value has been specified within the **Port** field.

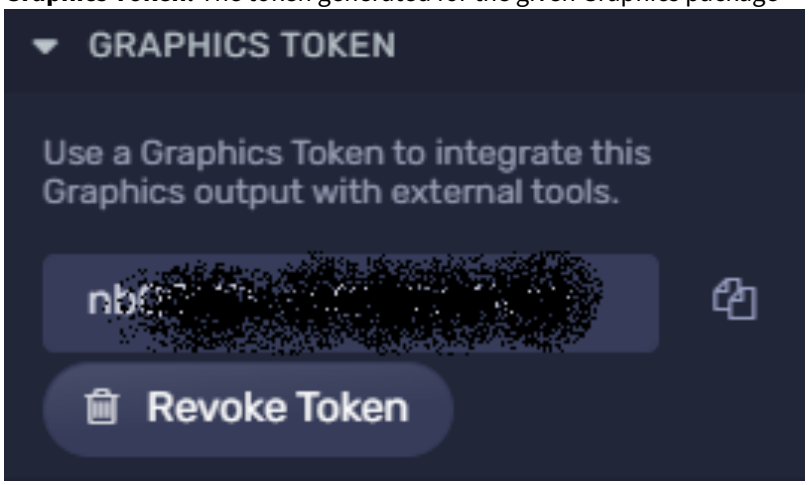
        - **Port:** TCP/IP port of the **Viz Engine**.
    - **Handler**
      These settings are only relevant when setting up Viz Multiplay video walls for use with Mosart.
        - **Arm & Fire:** Enables or disables the *Arm & Fire* workflow. **Note**: Only Viz Multiplay will utilize this workflow, not Viz Mosart.
        - **Publishing Point ID:** The ID of a publishing point where Viz Engine can retrieve video clips from. Example for a Viz One server: *urn:vme:default:PUBLISH.PublishingPoint:30*
        - **Publishing Point URI:** The URI of the defined publishing point. Example for a Viz One server: *https://my-vizone.internal/api/publish/publishingpoint/30*
        - **Superchannel:** When enabled, superchannel will be used instead of GFX channel for playout.

8.6.2 VIZ FLOWICS

Graphic Type = Flowics (Viz Flowics)



- **Graphics Controller properties:** The properties are the same as described in Common Graphic Controller Properties.
- **Flowics graphics package:** Contains the following properties in addition to Common Graphic Engine Properties:
  - **Configuration:**
    - **Graphics Token:** The token generated for the given Graphics package



  - **Description:**
    - **Name:** The name of the Graphics package

> **Info**: You need to install and configure the Viz Flowics MOS Gateway for this integration to operate.

Tip: Follow this Viz Flowics support article for more details on setting up this integration.

8.6.3 CHYRON

Graphic Type = Chyron (Chyron Lyric)

Chyron properties	
Active signal	
Delay (frames)	0
Distribute status	☐
Description	
Graphic Type	CHYRON
Id	0
Name	CHYRON LYRIC
Replace takeout logic	
Delay (frames)	0
Enable	☐

Chyron engine	
Behaviour	
Always Take In	☐
Continue Scene	☐
Configuration	
Channel	0
Encoding	
Host	
Port	0
Description	
Name	Engine 25
Handler	
Handler Id	25
Name	25

- **Chyron properties:** The properties are the same as described in Common Graphic Controller Properties.
- **Chyron engine:** The properties are the same as described in Common Graphic Engine Properties.

8.6.4 DEKO

Graphic Type = Deko

Deko properties	
Active signal	
Delay (frames)	0
Distribute status	☐
Description	
Graphic Type	DEKO
Id	0
Name	DEKO
Replace takeout logic	
Delay (frames)	0
Enable	☐

Deko engine	
Behaviour	
Always Take In	☐
Continue Scene	☐
Configuration	
Channel	0
Encoding	
Connection	
COM port	
Machine ID	0
Description	
Name	Engine 19
Handler	
Handler Id	19
Name	19

- **Deko properties:** The properties are the same as described in Common Graphic Controller Properties.
- **Deko engine:** Contains the following properties in addition to Common Graphic Engine Properties:
  - **Connection:**
    - **COM port:** Defines the serial port connected to the Deko engine.
    - **Machine ID:** Digit 0-9. May be used for multi-drop. Multi-drop lets you specify different fields or layers (TypeDeko) when more than one TypeDeko is connected to the Deko host. Set equal to `1` if there is only one.

> **Note:** For using a default take out scene in Deko, the setting *<item name="ClearScene" value=""/>* needs to be added to the IntelligentInterfaceConfiguration.xml file instead of the clear command. The value should be set to any scene that should be used for take out.

8.6.5 ORAD

Graphic Type = Orad (Orad Maestro)

Orad properties	
Active signal	
Delay (frames)	0
Distribute status	☐
Description	
Graphic Type	ORAD
Id	0
Name	ORAD MAESTRO
Replace takeout logic	
Delay (frames)	0
Enable	☐

Orad engine	
Behaviour	
Always Take In	☐
Continue Scene	☐
Configuration	
Channel	0
Encoding	
Host	
Port	0
Description	
Name	Engine 24
Handler	
Handler Id	24
Name	24

- **Orad properties:** The properties are the same as described in Common Graphic Controller Properties.
- **Orad engine:** The properties are the same as described in Common Graphic Engine Properties.

8.6.6 PIXEL POWER CC

Graphic Type = Pixel Power CC (Pixel Power Control Center)

Pixel Power Control Center properties	
Active signal	
Delay (frames)	0
Distribute status	☐
Description	
Graphic Type	PIXEL POWER CC
Id	0
Name	PIXEL POWER CC
Replace takeout logic	
Delay (frames)	0
Enable	☐

PixelPower CC engine	
Behaviour	
Always Take In	☐
Continue Scene	☐
Configuration	
Channels	0
Encoding	
Host	
Port	0
Description	
Name	Engine 23
Handler	
Handler Id	23
Name	23

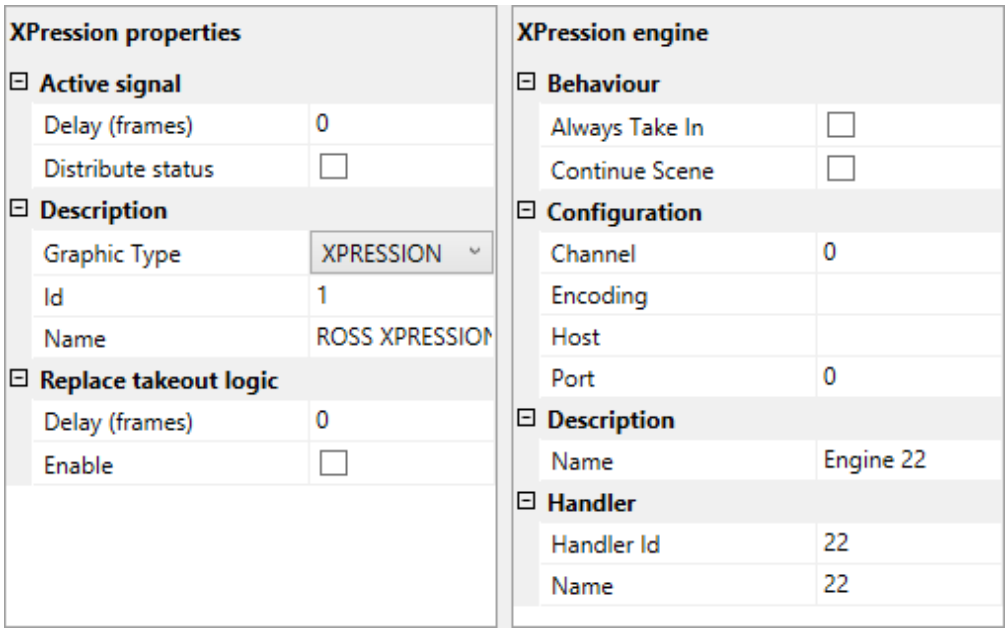
- **Pixel Power Control Center properties:** The properties are mostly the same as described in Common Graphic Controller Properties.

Copyright © 2026 Vizrt

- **Pixel Power CC engine:** The properties are the same as described in Common Graphic Engine Properties.
  - **Configuration**
    - **Channels:** Specifies the Pixel Power channels associated with the Pixel Power engine. Specify as a comma-separated list of integers.

8.6.7 XPRESSION

Graphic Type = Xpression (Ross Xpression)



- **XPression properties:** The properties are the same as described in Common Graphic Controller Properties.
- **XPression engine:** The properties are the same as described in Common Graphic Engine Properties.

8.6.8 BIGVIZ

Graphic Type BigViz is a BBC developed system that integrates with ENPS, Vizrt graphics, IBIS Playout, and Autoscript.

- **BigViz properties:** Controller connection properties are not used. All connectivity to BigViz is configured as part of the Viz Engine configuration.
- **BigViz engine:** The following properties are used only (all other properties related to Viz Graphics configuration are not used by BigViz):
  - **Host:** Hostname or IP address of BigViz service.
  - **Port:** TCP/IP port of remote BigViz service. Default port is `8165`.
  - **Client ID:** Identifies the Mosart server. Must match the corresponding BigViz configuration. Default: *mosart*.

8.7 Named Overlay Graphics

Named Overlay Graphics enables overlay graphics to be fired from within a template or from a control command (keyboard shortcut).

Named overlays are essentially static graphics, for example a clock or bug, that you want to be available across all rundowns, rather than needing to put them into the plan.

All named overlay graphics must be placed within the file *NamedOverlayGraphics.xml*.

This section contains:

- Working with Named Overlay Graphics
- NRCS-based Systems
- Viz Flowics-based Systems

8.7.1 Working with Named Overlay Graphics

When to use

Overlay graphics are fired from within a template or control command (as a keyboard shortcut).

- Named overlay graphics are treated as constants and not related to any specific rundown
- Named overlay graphics can be used to extract information from the current rundown.

Prerequisites

All named overlay graphics must be placed within a single XML file named *NamedOverlayGraphics.xml*. This file must be placed as part of the configuration files for the system, normally *%programdata%\Mosart Medialab\ConfigurationFiles*.

NamedOverlayGraphics.xml contains all overlay graphics to be accessible from templates.

The overlay graphics within this file have the following properties:

- The *slug* attribute is used as the 'name' of the overlay graphic (and used in a template to refer to the overlay graphic).
- The *slug* has to be unique within the NamedOverlayGraphics.xml file.
- The content of each overlay graphic shall be identical to the overlay graphic representation in Manus Administrator files. In most circumstances only a small set of the attributes and elements are necessary.
- The *templatetype* attribute is used to identify the behaviour of the overlay graphic and should be according to the lower third mapping found in *newsroomsettings.xml*.
  Default mapping is *AUTOOUT*.
- Setting **templatetype** to *STORYSTART* takes the corresponding overlay graphic on story transitions, when the first item or background of the story is taken.
  Only *one* overlay graphic can be used for this purpose.

8.7.2 NRCS-based Systems

Info: If working with Flowics graphics, see Viz Flowics-based Systems below.

Obtaining a valid overlay graphic as a Named Overlay candidate

To obtain candidates for named overlays

1. Forward the graphics you want to be part of named overlay as ordinary overlay graphics, by assigning them in the NRCS.
   Viz Mosart now reads these as part of overlays in a rundown.
2. Verify that the named overlay candidates appear in Overlay Graphics Interface left panel. This is where all active overlay graphics in the rundown are displayed.
3. Select a named overlay candidate in the Overlay Graphics Interface left panel and press the **Info** button beneath.
4. An XML representation of the selected candidate is displayed.
5. Copy this XML to the clipboard.
6. Paste the clipboard content into the named overlay XML file, *NamedOverlayGraphics.xml*.
7. (Optional) Remove not-required fields, as described in section Named Overlay Graphics.

NamedOverlayGraphics.xml Example

Below is an example, showing two named overlay graphics. One to be fired at every story transition (*MosartStoryStart*) and one that could be triggered from a template (*StrapsOff*).

This example is created for BIGVIZ which makes use only of the <objParams> field.

```
<?xml version="1.0" encoding="utf-8" ?>
<items>
<!--Overlay graphic used to inform BigViz about story start -->
<item slug="MosartStoryStart" templatetype="STORYSTART" in="0" dur="25"
mosid="BIGTED.W1.BBC.MOS" objid="BIGTED">
    <content>
     <storyItem>
       <objParams>MOSART STORY START</objParams>
     </storyItem>
    </content>
</item>
<!--Overlay graphic used to inform BigViz to take all overlay graphics off air -->
<item slug="StrapsOff" templatetype="AUTOOUT-DSK" in="0" dur="25"
     mosid="BIGTED.W1.BBC.MOS" objid="BIGTED">
   <content>
     <storyItem>
       <objParams>STRAPS OFF</objParams>
     </storyItem>
   </content>
</item>
```

```
</items>
```

NRCS Placeholders

Named overlay graphics supports placeholders for NRCS information. The following example shows a modified *MosartStoryStart* named overlay graphic that makes use of placeholders:

```
<item slug="MosartStoryStart" templatetype="STORYSTART" in="0" dur="25"
mosid="BIGTED.W1.BBC.MOS" objid="BIGTED">
   <content>
     <storyItem>
       <roID>{story:roid}</roID>
       <storyID>{story:id}</storyID>
       <storySlug>{story:slug}</storySlug>
       <objParams>MOSART STORY START</objParams>
     </storyItem>
   </content>
 </item>
```

Syntax: `{[story|item]:[story_value|item_value|xpath]}`

Where:

- **story:** Obtains information from the current story:
  - **roid:** MOS rundown identity.
  - **id:** MOS story identity.
  - **slug:** MOS story slug.
  - **xpath:** Arbitrary xpath from Manus Administrator story element.
- **item:** Obtains information from the current story item:
  - **roid:** MOS rundown identity.
  - **id:** MOS story identity.
  - **slug:** MOS story slug.
  - **type:** Viz Mosart type.
  - **variant:** Viz Mosart variant.
  - **template:** Viz Mosart type + Viz Mosart variant.
  - **xpath:** Arbitrary xpath from Manus Administrator story item element.

Examples of valid placeholders

- **{story:roid}:** The rundown identity.
- **{story:id}:** MOS story identity.
- **{story:slug}:** MOS story slug.
- **{item:type}:** MOS story item, Viz Mosart template type.
- **{item:variant}:** MOS story item, Viz Mosart template variant.
- **{item:template}:** MOS story item, Viz Mosart template type+variant.
- **{item:@templatetype}:** xpath, Viz Mosart template variant.
- **{item:item[@type=100][1]/*/storyItem/objParams}:** xpath, picks legend from first (**[1]/*/**) lower third (**[@type=100]**) object (**objParams**) within a story item (**storyItem**).

Available Actions

It is possible to trigger defined actions within the Overlay Graphics Interface when taking a named overlay graphic. The actions are specified within an action list for the overlay graphic. A typical use case is when the graphics system requires specific graphic properties to be supplied, for an item to be taken out.

A sample action list is defined in the XML as follows:

```
<item slug="LOGO OFF" templatetype="AUTOOUT-DSK" in="0" dur="12" >
   <actions>
     <action name="takeOut" value="[last¦lastLocator¦lastManual|regex]" />
     <action name="clear" />
   </actions>
   <content>
   </content>
</item>
```

The following actions are available:

- **takeout:** Takes out any stored on-air graphic matching the criteria specified in the value field:
    - **last:** Takes the last taken graphic out.
    - **lastLocator:** Takes the last locator graphic out.
    - **lastManual:** Takes the last manual graphic out.
    - **regex:** Takes out all graphics with the slug matching the given regular expression. I.E regex = “LIVE.*” will take out all graphics with slug starting with “LIVE”. Regular expressions follows the Microsoft .net regular expression language.
- **clear:** Takes out all on-air graphics.

XML Fields

Example of complete lower third graphics XML

An example graphics XML extracted from a rundown:

```
<item type="100" slug="In:00:00/LIVE// Mosart=L|00:00|B" source="1"
index="100_NWATVNT1;P_ATVNEWSWF_HOLD MOSART
TRAINING;AFD78EF5-3D19-48F1-860BB9B2B53A679F_NWATVNT1;P_ATVNEWSWF_HOLD MOSART
TRAININGR_AFD78EF5-3D19-48F1-860BB9B2B53A679F;21115F3E-B25A-46AD-89620E63F8077CCF_2"
idref="2" templatetype="BACKGROUNDEND-DSK" status="0" error="0" in="0" dur="500"
pin="0" pdur="500" rdur="0" externaleffect="" intimeline="true" date_0=""
accessory="False" static="false" endfrase="" rundown="NWATVNT1;P_ATVNEWS\W\F_HOLD
MOSART TRAINING;AFD78EF5-3D19-48F1-860BB9B2B53A679F"
storyid="NWATVNT1;P_ATVNEWS\W\F_HOLD MOSART
TRAINING\R_AFD78EF5-3D19-48F1-860BB9B2B53A679F;21115F3E-B25A-46AD-89620E63F8077CCF"
typetext="" mosid="PILOT.ATV.TEN.MOS" objid="1863809" ismoselement="true"
use_graphics_id="true" graphics_id="1863809" handler_name="DSK"
graphics_out_on="BACKGROUNDEND" description="(DSK) - In:00:00/LIVE// Mosart=L|00:00|
B" owner="">
```

```
   <fields>
     <field name="graphics_description" fieldtype="TEXT" value="In:00:00/LIVE//
Mosart=L|00:00|B" />
     <field name="graphics_id" fieldtype="TEXT" value="1863809" />
     <field name="tc_dur" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:20" />
     <field name="continuecount" value="-1" fieldtype="TEXT" />
     <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss" default="00:00"
value="00:00" />
   </fields>
</item>
```

Required fields: `slug, templatetype, in, dur, use_graphics_id, graphics_id, handler_name, graphics_out_on, description`.

Extracted XML example

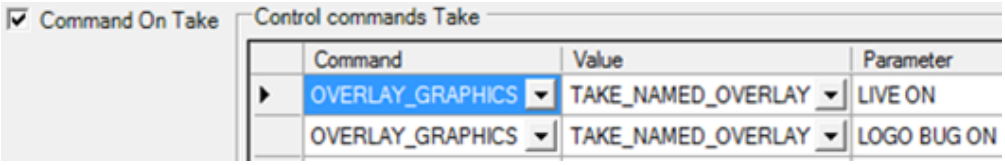
For the required fields extracted:

- The name used to recall the named overlay graphic is *LIVE ON*.
- Note that we have changed the *slug* and *description*.

```
<items>
<!--LIVE ON -->
    <item slug="LIVE ON" in="0" dur="125" use_graphics_id="true"
graphics_id="1863809" handler_name="DSK" graphics_out_on="BACKGROUNDEND"
description="(DSK) - LIVE ON"/>
</items>
```

To add a named overlay graphic to a template

- SLUG is where we name the overlay for our use – and we can now recall the overlay graphic from a template as shown below:



8.7.3 Viz Flowics-based Systems

Info: If working with NRCS graphics, see NRCS-based Systems above.

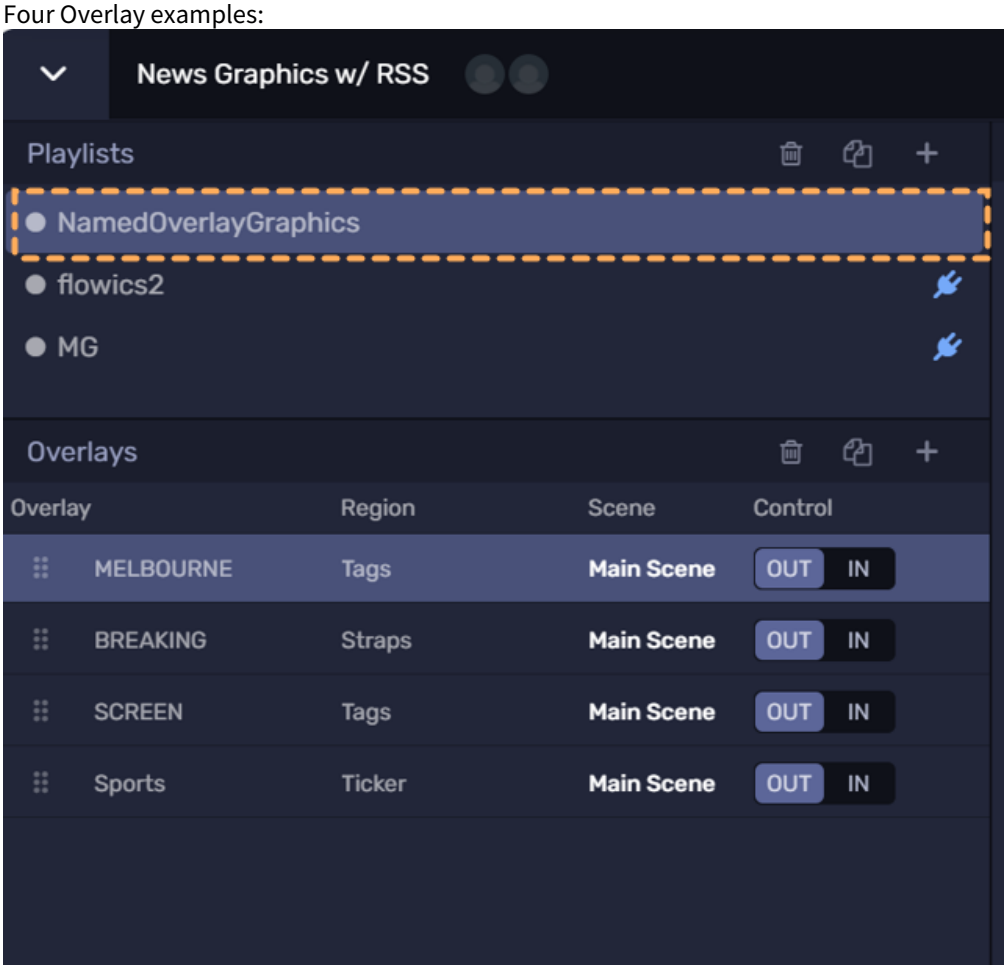
On the Flowics Rundown Control page, you will manually create a non MOS-controlled playlist in the graphics package.

Your manually created playlist will persist in the Flowics package playlists list, until deleted.

Obtaining a valid overlay graphic as a Named Overlay candidate

To obtain Flowics candidates for named overlays

1. Log-in to your Flowics system.
2. In the **Playlist** panel, for rundown control of your Flowics graphics package, create a playlist called *NamedOverlayGraphics*.
3. In the **Overlays** panel, manually add overlays from the package.
   Four Overlay examples:



4. Rename the slugs that will be referenced by Viz Mosart.

> **Info**: Names in the list must be *unique* as these names will be referenced by Viz Mosart and must match what is in the *NamedOverlayGraphics.xml* file.

5. On the Mosart server, locate the file *NamedOverlayGraphics.xml* (usually in the *channeltemplates* folder).
6. For each item row.
   a. Edit the value of item **slug** to match the name in your Flowics playlist.
   b. Change the value of **mosid** to *flowics.com.mos*
   c. Change the value of **graphics_id** to *flowics*
7. Restart Manus Administrator.
8. Restart the Overlay Graphics Interface (OGI).

8.8 Managing Overlay Graphics Controllers and Engines

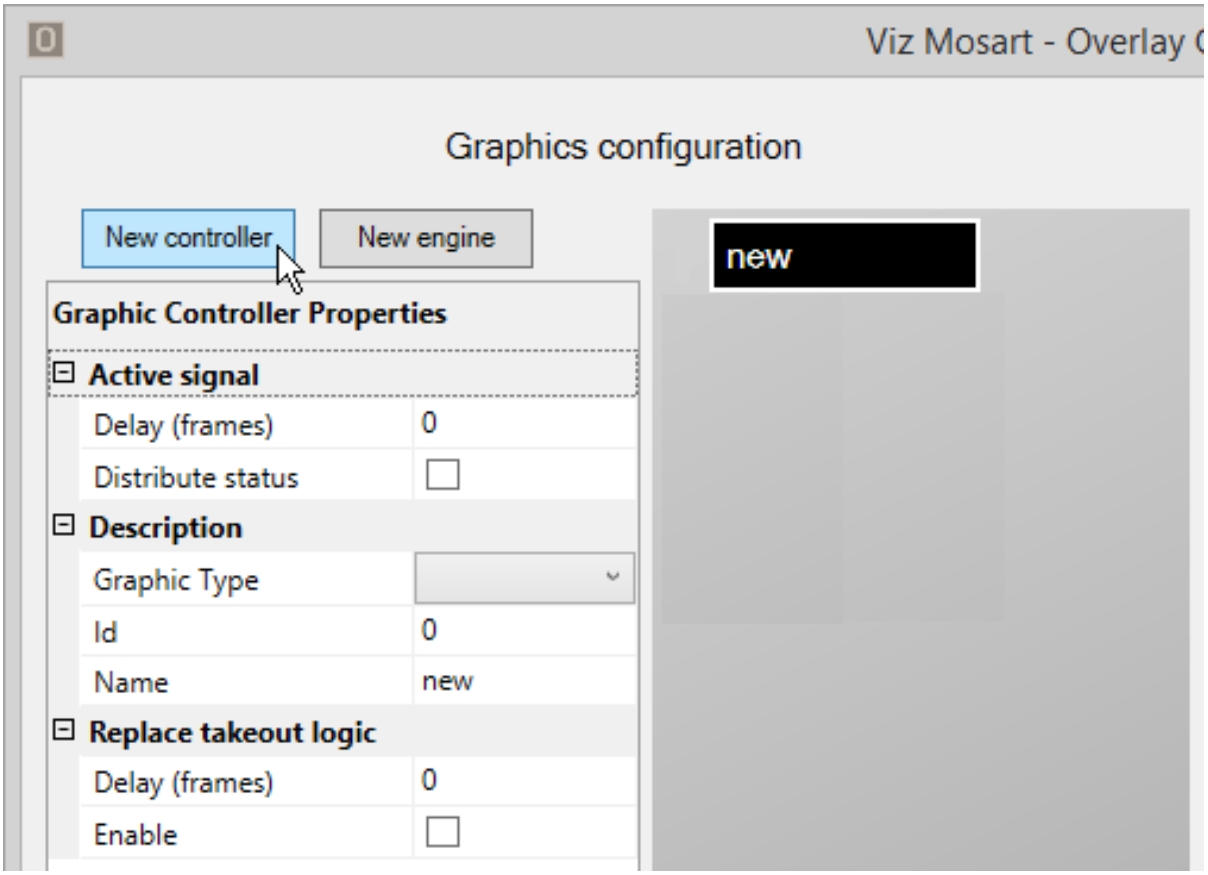
8.8.1 Overlay Graphics Controllers and Engines

- Working with Controllers
- Working with Engines
- Common Graphic Controller Properties
- Common Graphic Engine Properties

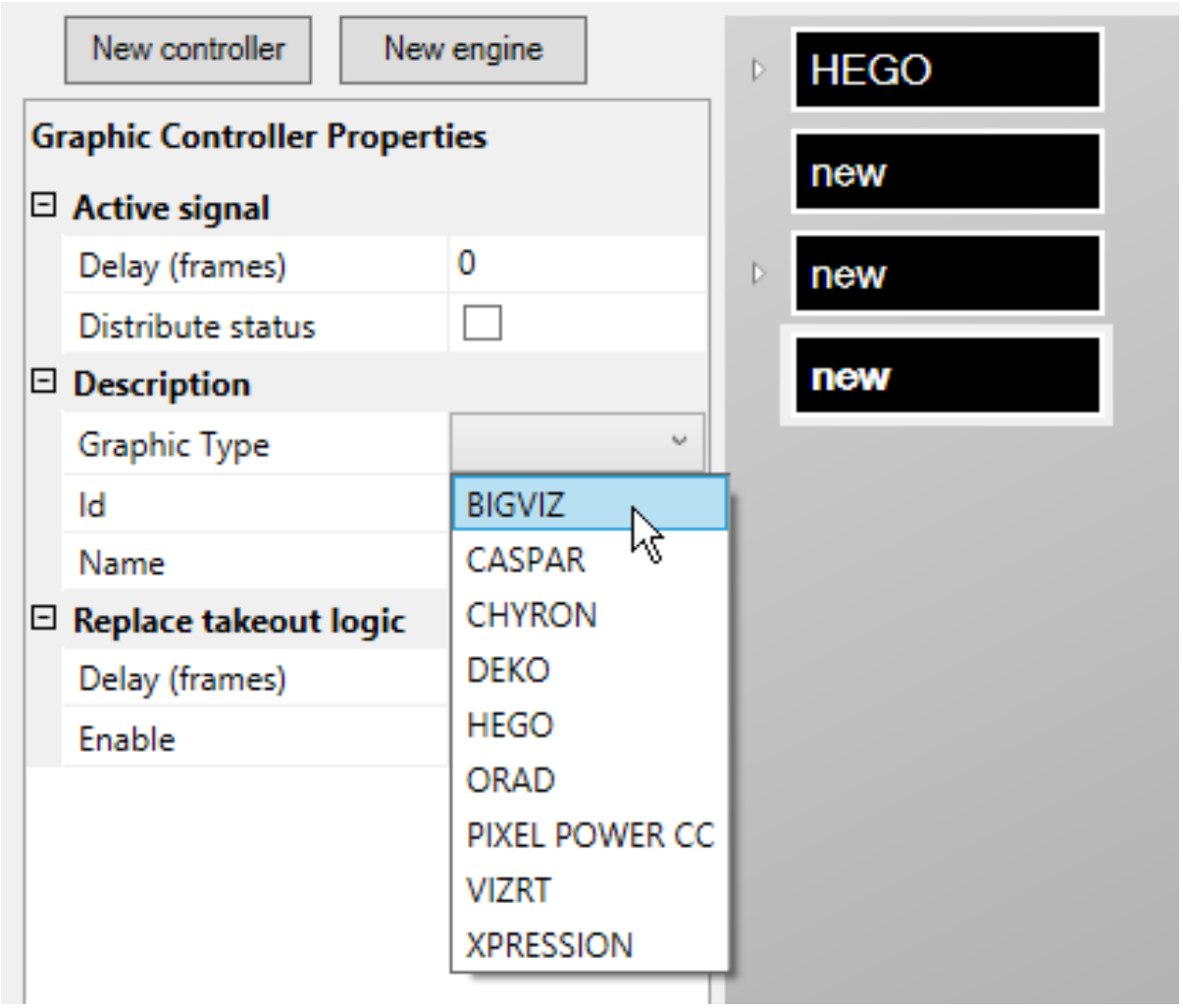
Working with Controllers

To add a new Controller

1. Open the Overlay Graphics Configuration window by going to **Settings > Properties**.
2. Click **New controller**.



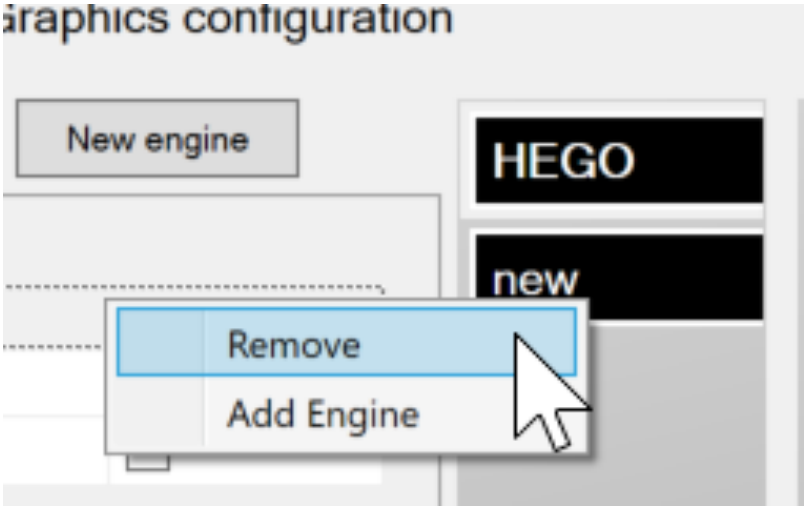
3. In the **Graphic Type** drop-down, select the type of graphics system you are using (see Overlay Graphics Types).



4. Set additional controller properties according to the selected **Graphic Type**.
5. Optionally, add graphics profiles via the Graphics Profiles Tab. Only applicable for some graphics systems.
6. To add engines to the controller, see To Add a New Engine below.

To remove a Controller

1. Open the Overlay Graphics Configuration window by navigating to **Settings > Properties**.
2. From the list of controllers, select the black box of the controller you want to remove.

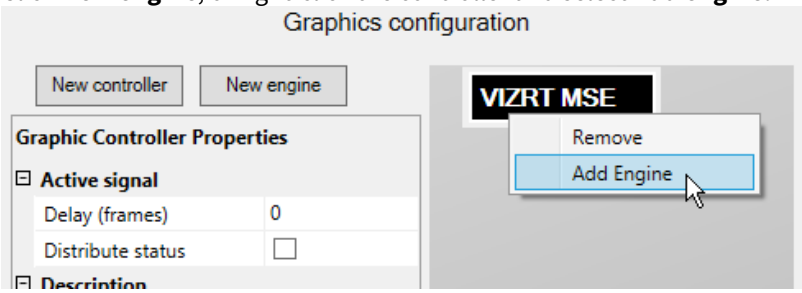


3. Right-click and select **Remove**.

Working with Engines

To add a new Engine

1. Open the Overlay Graphics Configuration window by navigating to **Settings > Properties**.
2. From the list of controllers, select the black box of the Controller you want to use.
3. Click **New engine**, or right-click the controller and select **Add engine.**



4. Set the engine properties according to the selected graphic type (See Overlay Graphics Types).
5. Optionally, set actions for selected system events using the Actions Tab (Event and Action Rules).

To remove an Engine

1. Open the Overlay Graphics Configuration window by navigating to **Settings > Properties**.
2. From the list of engines, select the black box of the engine you want to remove.
3. Right-click and select **Remove**.

Common Graphic Controller Properties

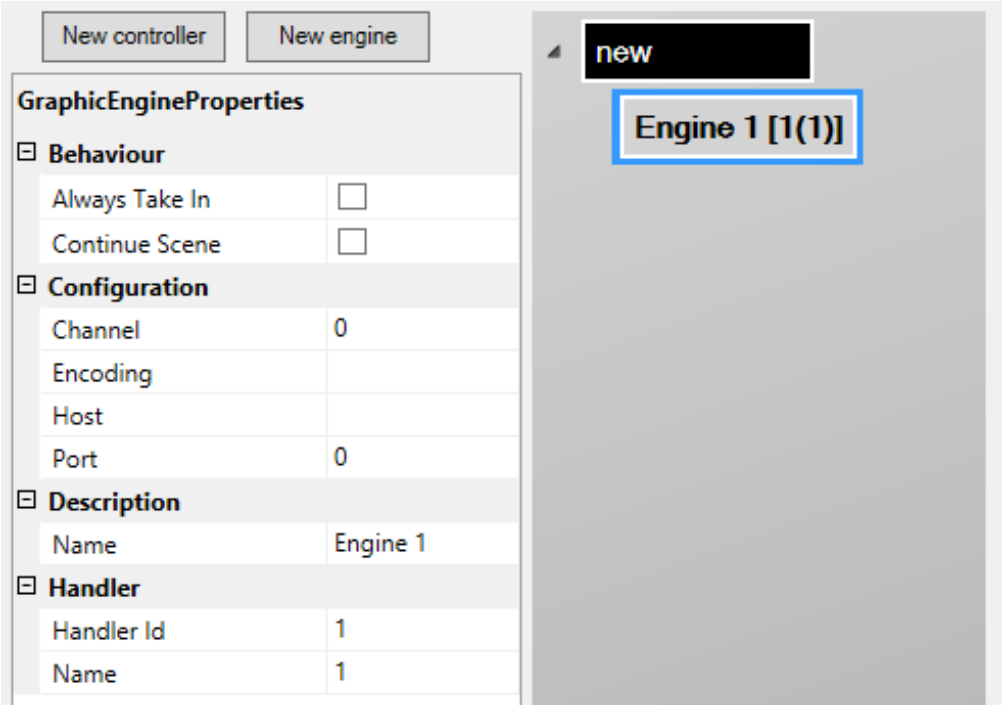
The Graphic Controller properties that are available depend on the graphic type of the controller. The controller properties common to most graphics types are described below.

> **Note**: Additional properties specific to each graphic type are described in section Overlay Graphics Types.

- **Active signal:** The active status signal is sent to AV Automation containing on air status of a graphics engine.
  - **Distribute status:** When set to t*rue*, reports the on air/usage status of an engine to AV Automation. This status message is sent to the AV Automation log area, the Manus Administrator console and the Overlay Graphics log area.
  - **Delay (frames):** Sets the delay of the graphics active status changed event signal. Using a negative value for the delay disables the functionality (**Distribute status** is *false*).
    Upon receiving the signal, AV Automation performs the following actions:
    - Sets the DSK if this is enabled (**Control DSK from OverlayInterface** is checked on the **Switcher preferences** page and the switcher can set *DSK*).
    - Executes direct take templates configured on the **Switcher preferences** page in **Overlay DirectTake control** (see section AV Automation Devices Vision Mixer).
    - Cuing the graphics in certain conditions (the rundown story item is in preview, the channel template used has a graphics device setup, the engine is not in use, the channel is not on air).
- **Description**
  - **Graphic Type:** Selects the brand and type of graphics.
  - **Id:** Identifies the graphics controller (unique number ID). Shall be unique among all defined controllers.
  - **Name:** Defines the name of the controller (optional).
- **Replace takeout logic:** Reads out the transition logic context of elements to either suppress take out or re-take items. Take outs are suppressed if another item with the same context was taken before the scheduled for take out action. Take outs are replaced with re-takes if the item scheduled for take out replaced an item with the same context. Only for Vizrt graphics with transition logic.
  - **Delay (frames):** Delay when checking for duplicate ‘take in’ commands in the next element.
  - **Enable:** Use transition logic aware take in/out commands.

Note: See also Graphics Profiles Tab for other controller settings.

Common Graphic Engine Properties



The **Graphic Engine Properties** that are available depend on the graphic type of the controller. The engine properties common to most graphics types are described below.

The additional properties for each Graphic Type are described in Overlay Graphics Types.

- **Behavior**
  - **Always Take In:** Turns off the logic that does not take in a graphic element that is already taken and no other graphic element has been taken since.
  - **Continue Scene:** Use continue for take out of scene based graphics.
- **Configuration**
  - **Channel:** Defines the output channel from the graphics system. Set to *0* when not used for takes.
  - **Encoding:** Encoding to use on text. *UTF-8* or *UTF-16*. Default: *UTF-8*.
  - **Host:** Defines the hostname or IP address of the graphics engine. See also descriptions for the various graphics types.
  - **Port:** Defines the TCP port to use when connecting to the graphics system.
- **Description**
  - **Name:** The name of the engine.
- **Handler:** ID and name of the handler. Note that if handler ID is *changed* and the engine is assigned to a destination, then the link must be removed and engine re-assigned.
  - **Handler Id:** Auto-generated ID.
  - **Name:** The name of the handler (auto-generated).

Note: See also:

- Section Graphics under *AV Automation Device Properties.*

9 Show Design and Template Editor

Central to reliable and error-free productions is the Viz Mosart concept of *templates*. A template is pre-programed before the production during *show design*, with help of the Viz Mosart utility, *Template Editor*.

Each template performs a device-specific operation, like studio camera selection or playing a specific video clip from a video server.

This section introduces some examples of how complex studio device operations can be programed onto a template, and walks you through show design, where the template operations are programmatically chained together in a Viz Mosart *rundown*.

- Overview
- Template Examples
- Working with Templates
- Other Template Functionality
- Working with Template Sets

9.1 Overview

Behind each operation, as displayed on the timeline in Viz Mosart's Rundown view, is a Viz Mosart template.

A Viz Mosart template is the key element of controlling external devices and events handled by Viz Mosart. A template can send simultaneous control commands to multiple studio devices.

Typically, the actions defined on a template are either:

- **Automatic**
  Either invoked from another template or at a given time as part of the rundown
- **Preconfigured**
  Configured in the newsroom control system (NRCS or NCS) as part of a rundown, and then manually executed by the Viz Mosart operator
- **Manual**
  The operator initiates the template-driven operation, usually from a keyboard shortcut or touch panel.

Some essential template concepts are introduced below:

- Template Sets
- Working with Templates
- Template Terms and Definitions

9.1.1 Template Sets

Templates are organized in *template sets*. Typically, there will be a template set for each show/program. As Viz Mosart runs, one template set will be *current*, meaning templates that will be executed during the show are drawn from this set.
An alternative set can be selected as the new current set, see Current template set below. The templates that comprise a show can be arranged in two ways:

- Standalone
- Hierarchical

Standalone

Traditionally, Viz Mosart template sets are treated individually, with no connection to each other.

For example, a template set is created for each show or program. This is usually combined with automatically selecting the template set through the associated NRCS rundown. In addition, startup commands for preparing the studio (lights, sound and other devices) for running the show are included in the template set.

When creating a template set for a *new* program, it is based upon a copy of an existing template set, with small adjustments according to the functionality needed by the program.

When this new template set is created (by copying an existing set), a copy of *all* templates is made, and no references are made between the original and copied templates. Each template set is classified as ‘standalone’.

Disadvantages with this approach:

- Many duplicates are created. Normally only a few of the templates in the set need to be adjusted for meeting the needs of a new program.

- With an ever-growing bank of templates being created for each new show, the amount of stored templates in Viz Mosart can exceed system norms. This results in long response times when saving templates and increases the memory footprint of both the Mosart Server and Viz Mosart client workstations.
- Maintenance of templates becomes cumbersome and error prone. When a change is made in a template that is common for all template sets, the same change has to be applied individually to every template set. This is a time consuming task across numerous template sets.

Hierarchical

An alternative approach to Standalone is *hierarchical* template sets. This method also enables a new template set to be created, based on the logic of an existing template set. However, to begin with, *no* templates are copied from the existing template set.
Only when a change is made to an existing template, or when a new template created, is a template created and stored in the new template set.
The hierarchical approach can be adopted with all existing (standalone) template sets, to optimize machine capacity and introduce a more efficient workflow.

Referring to the background discussion in Working with Template Sets above, when you copy a template set, it is more efficient to use a hierarchical approach. These operations are usually performed by a system administrator, as outlined in Creating Hierarchic Template Sets below.

Direct Takes

In addition to standard show-related template sets, there is a special template set called *Directtakes*.

Unlike standard templates, templates in this set are *not* intended to be executed as part of the rundown.
Instead, they are usually manually executed during a running show *immediately* in an ad-hoc fashion.

Often a unique numeric value is assigned to each direct take. The operator enters executes the direct take template by entering its *number* on a numeric keypad.

9.1.2 Working with Templates

You create and edit individual templates and template sets (standalone and hierarchic) in the AV Automation application, **Template Editor**.

Creating a Hierarchic Template Set

- Templates are created and customized during the 'show design' phase of a Viz Mosart system setup.

This is a specialist operation, described in section Working with Template Sets (this document can also be browsed from Viz Mosart UI menu bar **Help > Viz Mosart Administrator Guide**).

Templates in the AV Automation Application

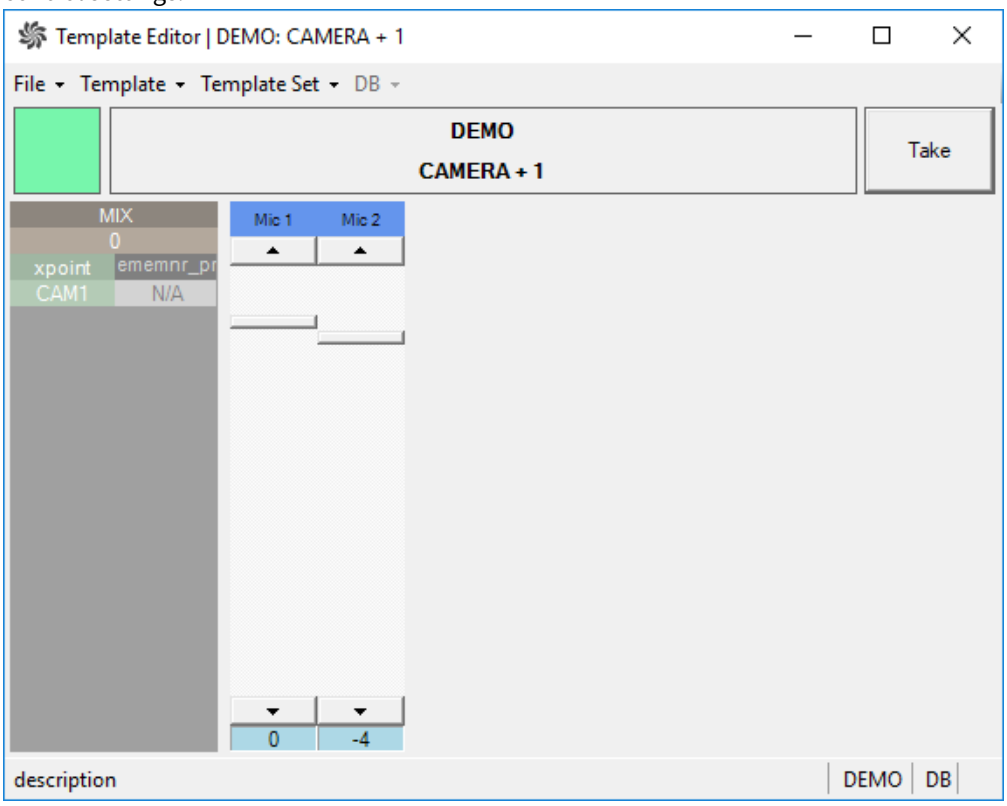
To create and modify show templates

- With a combination of mouse-over and contextual menus, an administrator can create and modify the templates that comprise a Viz Mosart-driven show.

This is presented in more detail later in this section.

To browse your show templates

- From the AV Automation console program (running on the Viz Mosart server), navigate to **Main menu > Devices > Template editor**.
  Templates displayed in Template Editor provide detailed information about their origin, purpose and control settings.



9.1.3 Template Terms and Definitions

Some common terminology used in template design and execution:

- **Base template set:** The start point for a show design.
  For example, if a template set *TS2* is based upon template set *TS1* then *TS1* is defined as the *base template set of TS2*. Programmatically, this relation can be written in short notation as *TS2* > *TS1*
- **Current template set:** The active template set in Viz Mosart. Changing the current template set is normally done from either:
  - The Viz Mosart client, from the **Studio setups** menu.
  - The NRCS (for example, ENPS or iNEWS) using the Viz Mosart control command [STUDIO SETUP]
  - The Viz Mosart server application Av Automation, from the the **Studio setups** menu.
- **Directtakes:** A special template set, where templates in this set are intended to be manually executed in an ad-hoc fashion.
- **Empty template:** An undefined, non-operative template. An empty template does nothing when executed. Note that empty templates are *not* displayed in the Viz Mosart client.

- **Gallery:** Corresponds to a Viz Mosart installation. Typically a gallery has a dedicated Viz Mosart server, with configured utilities and a Viz Mosart client.
- **Inherited template set:** A template set based on another template set.
  For example, if *TS2* > *TS1* then *TS2* is classified as an inherited template set.
- **Missing template:** A template that does not exist in the current template set. The operator is alerted to a missing template in the Viz Mosart client, where it is displayed with r*ed* text.
- **Shared template set:** The template set shared among all Viz Mosart installations that are connected to the same Viz Mosart template database.
- **Show design:** The background task of creating a set of templates, in a logical and programmatically valid order, for controlling a scheduled, rundown-driven, studio transmission.
- **Stand alone template set:** A template set not based on any other template set. All template sets in a current installation of Viz Mosart are classified as *standalone*. Programmatically, this relation can be written in short notation as *TS1 > ()*
- **Template description:** The template properties common for all Viz Mosart installations in a shared template set. This description uniquely identifies and describes the template.
- **Template implementation:** Contains the behavior or device control aspects of the template. The template implementation is related to the Viz Mosart installation. For a *shared* template set, each Viz Mosart installation will implement the template set differently, according to local studio setup.
- **Template set:** Viz Mosart templates are always contained in a template set. There may exist multiple *versions* of the same template but in *different* template sets. A template is uniquely defined by its *type* and *variant name* within its corresponding template set.
- **Template type:** Template type or category. Viz Mosart has a predefined set of template types. Each are displayed with a distinct visual appearance in Viz Mosart.
  Examples of template types are CAMERA, PACKAGE, VOICEOVER, LIVE, GRAPHIC and DVE.
- **Template variant:** Unique identifier of a template for given template type within a template set.
  For example, *camera1* and *camera2* are two variants of a camera template.

9.2 Template Examples

Each major control room operation is driven by a unique template *type*. These operations are allocated to a *primary* template.

Examples of primary templates are *Camera*, *DVE*, *Package*, *Voiceover*, *Live*.

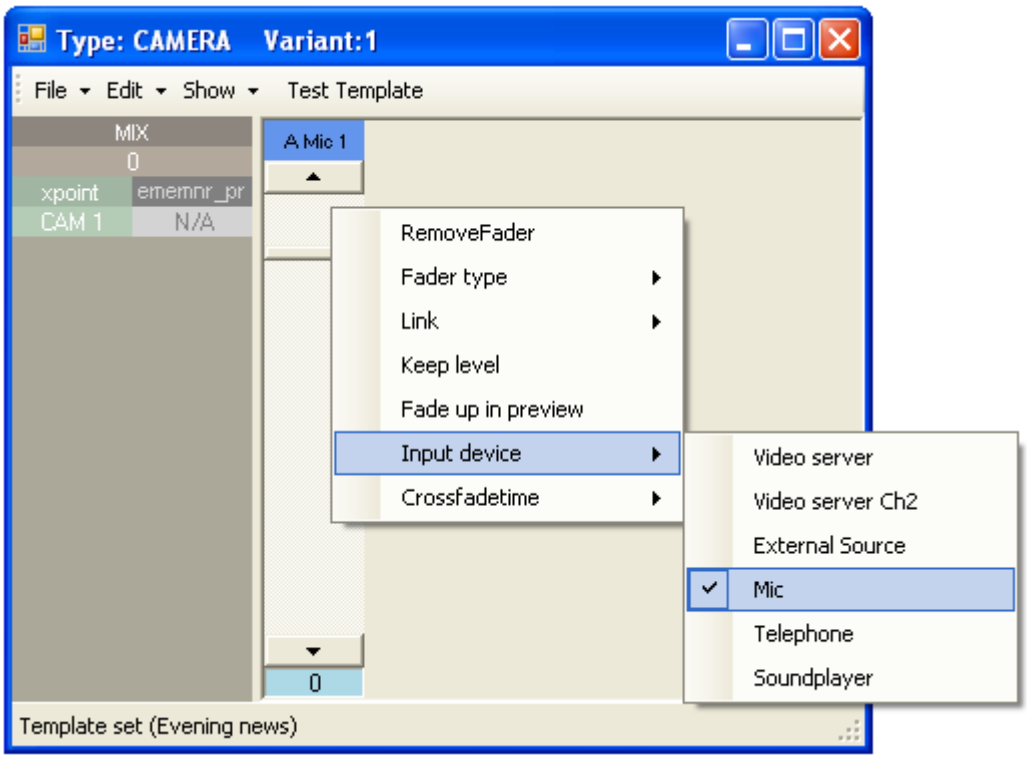
This section contains the following examples:

- Studio Camera
- Package: Video Clip with Full Sound
- Voice-over
- Live External Source
- DVE
- Graphic: Full Screen Graphics
- Accessory Templates
- Switcher Plugin

Full details of the operations displayed in the examples below are presented later in this section

9.2.1 Studio Camera

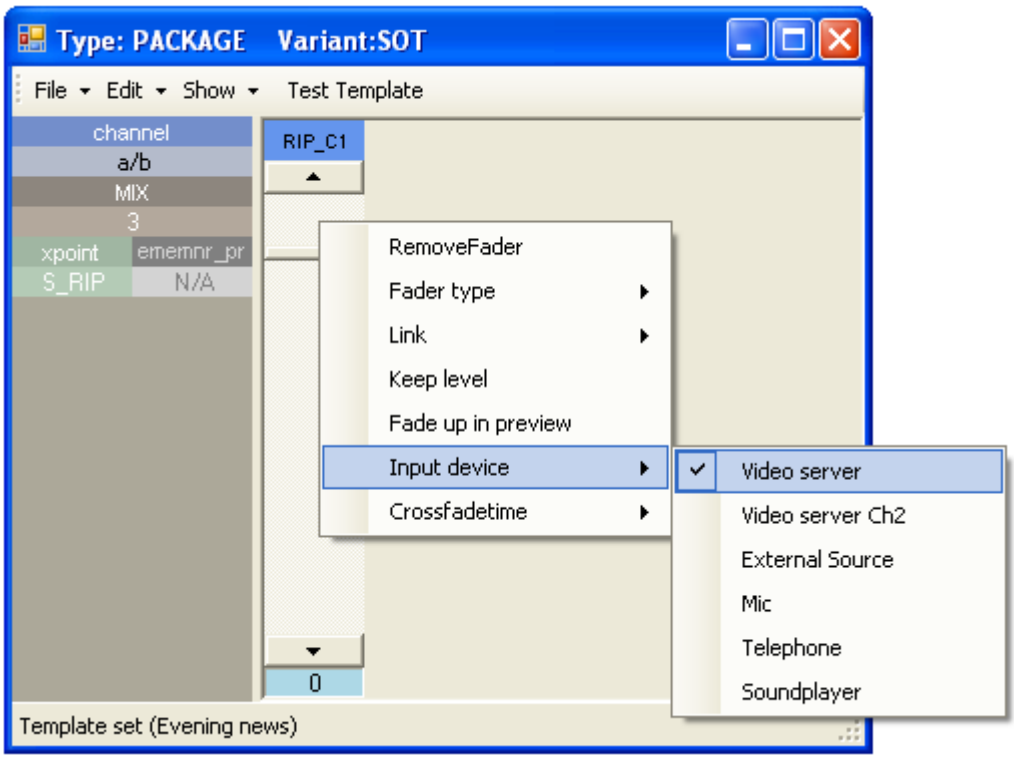
The template for a typical studio camera is based on the CAMERA type. The example below defines a standard camera 1 variant. An entry named CAM 1 in the video setup is selected at the PP bus cross point. Transition duration is `0` frames (hard cut) and E-mem recall is disabled. There is only one studio microphone, which fader is called "A Mic 1" in this template. Audio level is set to `0` dB, and the input device is a microphone. The input device type is important for the handling of audio in an ad lib sequence of full camera, DVE and full external source.



For our standard camera 2 variant, the only difference in the above setup is the PP bus cross point which is called CAM 2.

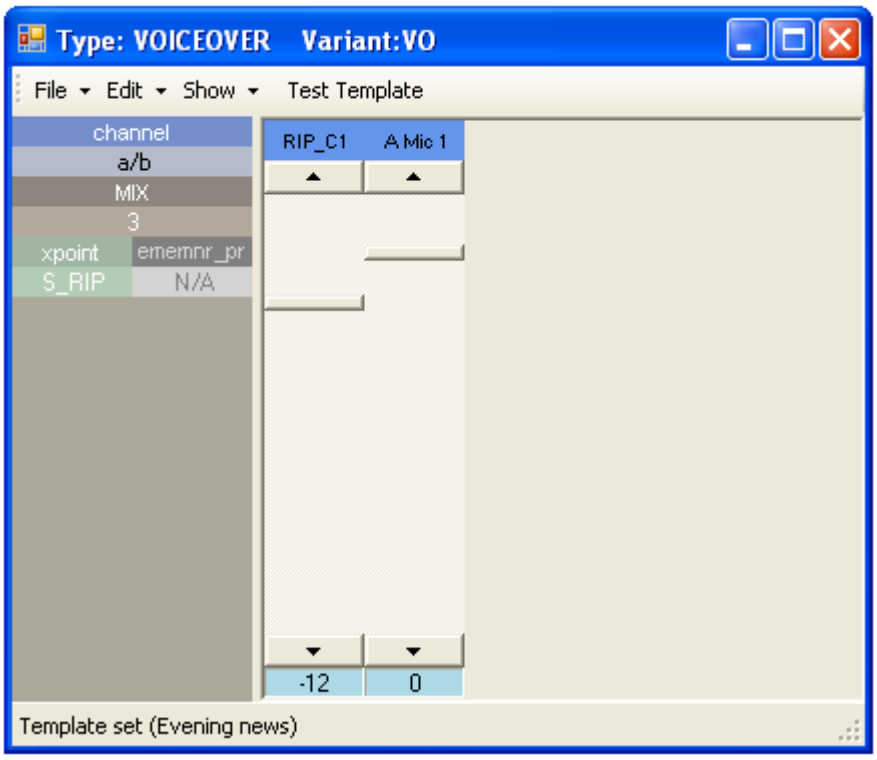
9.2.2 Package: Video Clip with Full Sound

As crosspoint on the vision mixer, we use a source called S_RIP, which means Viz Mosart uses the A/B roll and roll between the sources defined in the AV setup as *videoserverA* and *videoserverB*. We chose RIP_C1 as the audio source, but as long as A/B roll is enabled, any source defined as Video server follows the video source. The default transition is a three frame mix. The video channel control is also enabled and controls the clip. The server channel is automatically assigned within the automation.



9.2.3 Voice-over

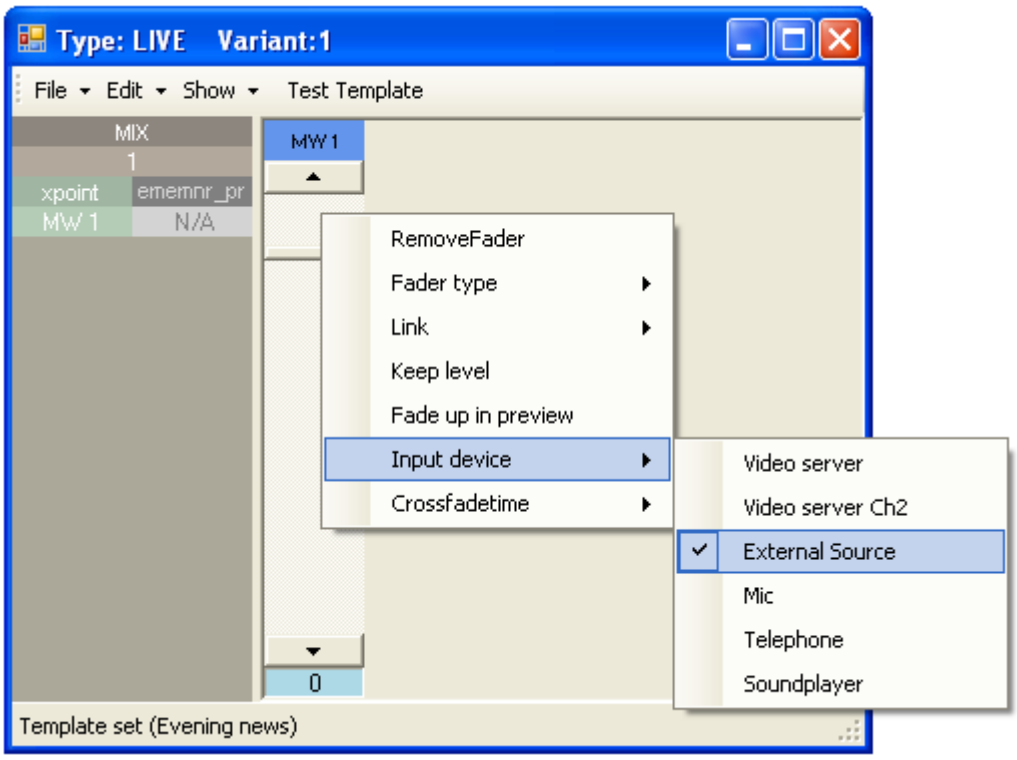
In this example, the variant VO is defined. Notice the difference from the two last examples and the two fader setups. A Mic 1 is microphone input type and RIP_C1 is video server type.



A second sound level can be set for each fader by using the **SHIFT** key over the fader to set the second level. This second sound level for all faders is used as the start level for the voice over sound bite type. The same special variant logic for video clip with full sound applies for the voice-over type.

9.2.4 Live External Source

Live external source variants are typically defined by their corresponding vision and audio mixer inputs. In the example below, we define variant 1 as source MW 1 on both PP bus cross point and audio fader.



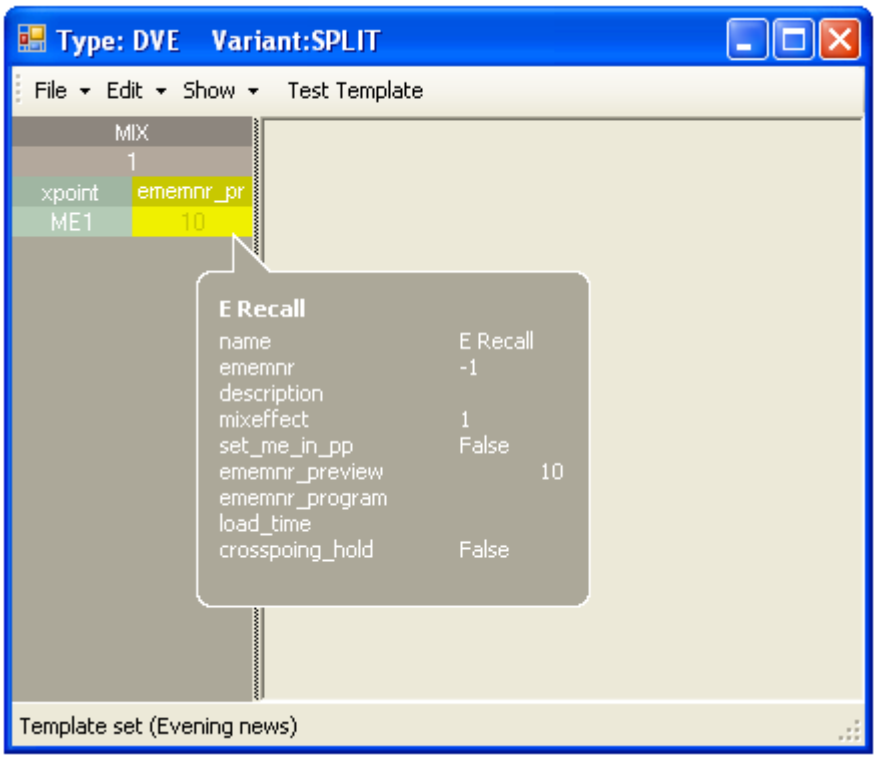
Additional variants 2 to 9 would typically use MW 2 to MW 9 as sources for both video and audio.

9.2.5 DVE

The example below describes four setup steps necessary to create a template that recalls a predefined E-mem (emem number *10*) defining a split screen DVE on M/E 1 on the vision mixer.

1. E-mem Recall Definition

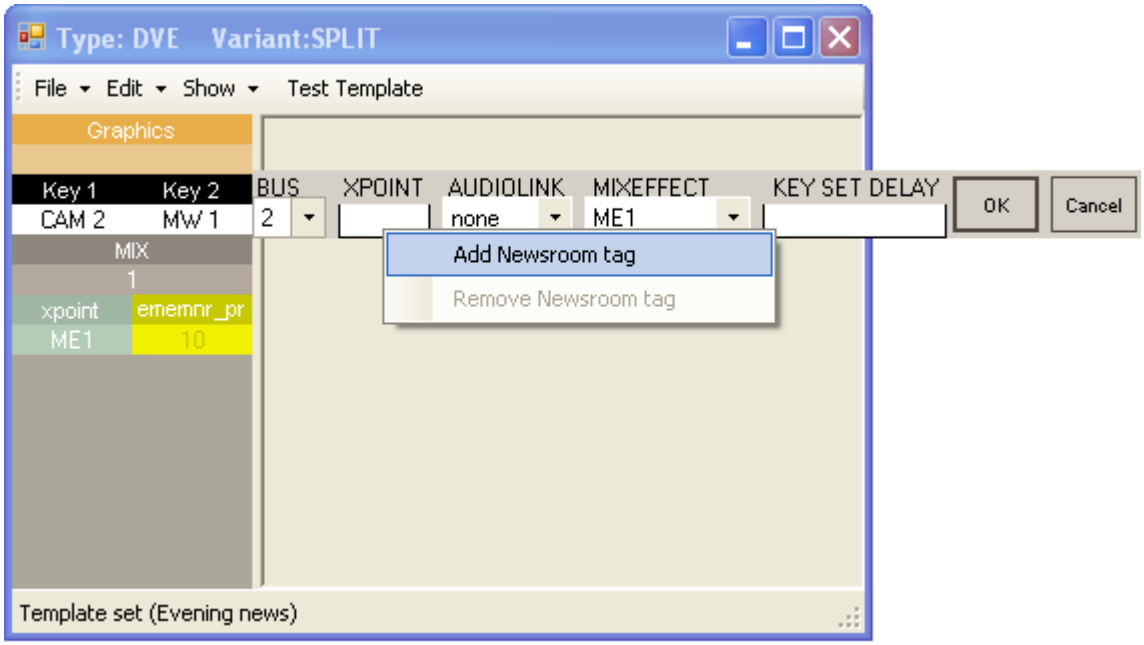
The vision mixer effect uses key 1 and key 2 on M/E 1 as left and right split windows.



The PP bus cross point is set to ME1 (program output of mixer effect). The E-mem recall control is defined to recall emem 10 when the template is taken to preview.

2. Key bus Delegation Setup

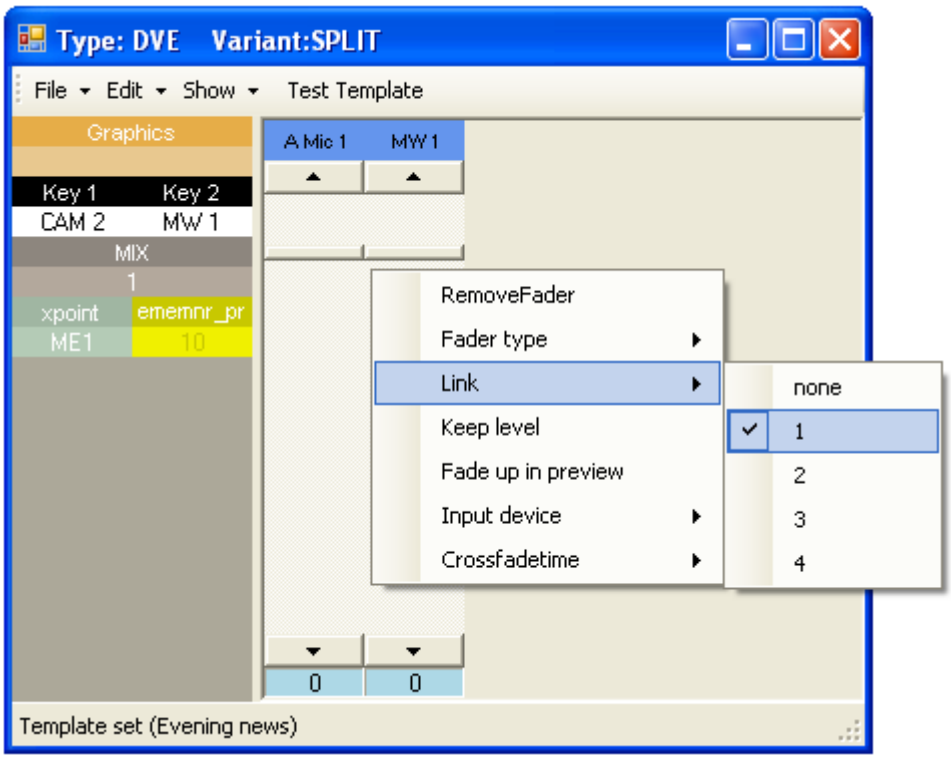
Default crosspoints are chosen (*CAM 2* and *MW 1*) and since a newsroom tag has been added for both keys, the crosspoints can be set from the newsroom system.
Right click over the XPOINT window to add a newsroom tag. In addition, the template defines a recall of a graphics background.



The second key bus delegation is shown. As a default we assign CAM 2 to key bus 1 and MW 1 to key bus 2 on the ME1 mixer effect bus.
(This assignment can be overridden from the NRCS story element by using the newsroom tag).

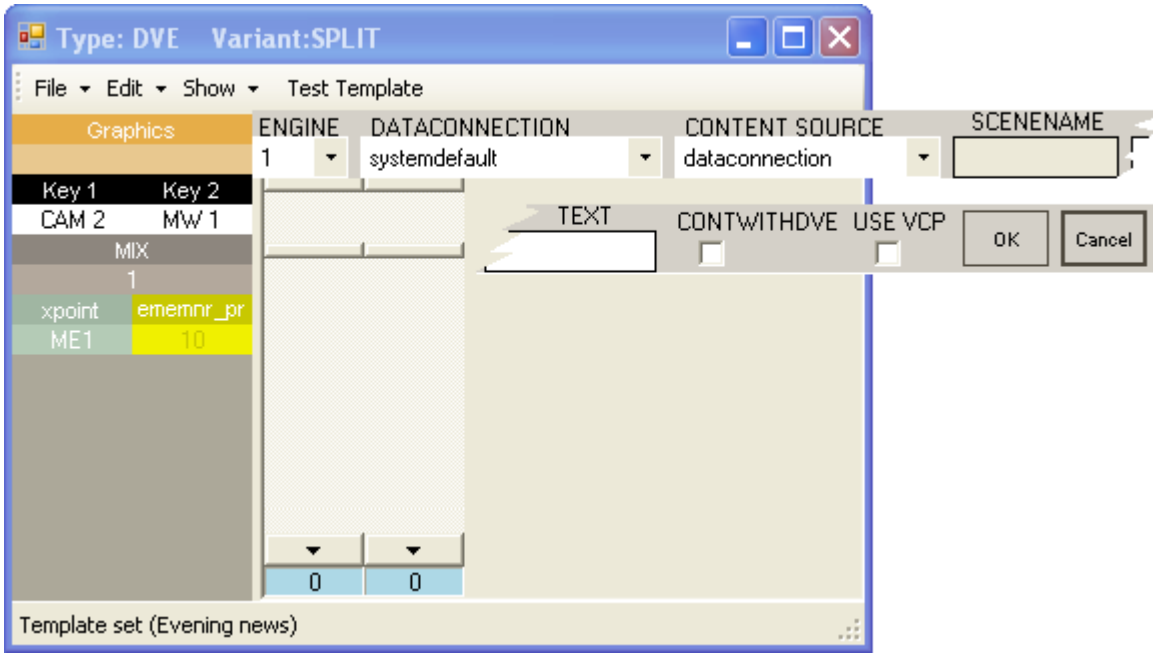
3. Audio Linking

The audio link is set to *1* for both the key 2 delegation and the second fader (MW 1).



When using the newsroom tag to set key 2 to a different source (for example *MW 3*), both video and audio delegation will follow:
This leads to the key 2 source being set to *MW 3* and the second audio fader being set to *MW 3*.

4. Graphics Engine Data Element Setup

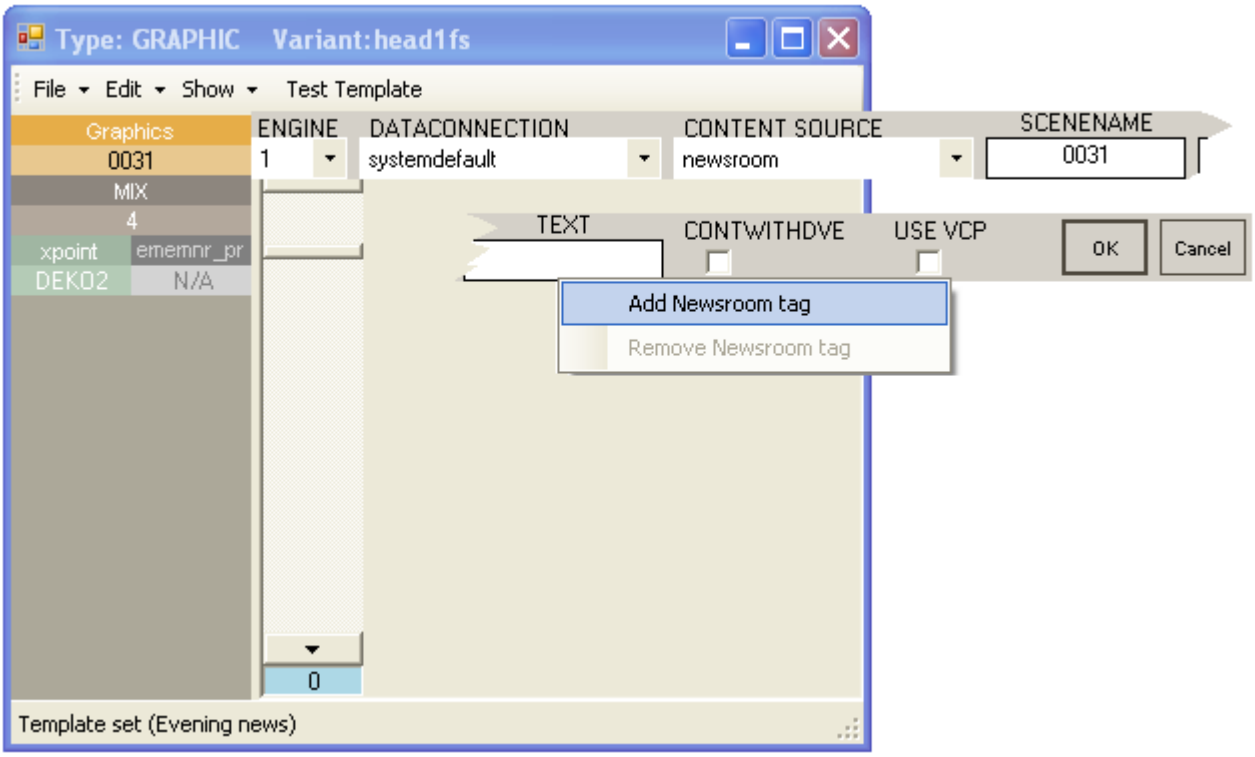


Setup of the *graphics controller* is described in the next section.

9.2.6 Graphic: Full Screen Graphics

The graphics controller in the **Template Editor** is vendor-specific, due to the lack of a generic control API. As a general feature, it can control multiple graphic engines. A typical controlled cluster could for example be, one for full screen graphics, one for studio wall graphics and additionally, an overlay graphics engine. For DVE effects, this can be a fixed element, while for telephone graphics or maps the element name and data are provided in the NCS story.

The controller's task is to cue graphics when taken to preview and run them when taken to program.



9.2.7 Accessory Templates

In this example you create an accessory template for driving content to a video wall driven by the Template Router feature in Viz Mosart. this feature merges new template-based instructions onto an existing one, enabling, as in this example, an asset to be diverted to the video wall. You need a separate accessory template for each wall that will be controlled. The accessory template can contain settings for cross points, video server port, graphics engine and aux.

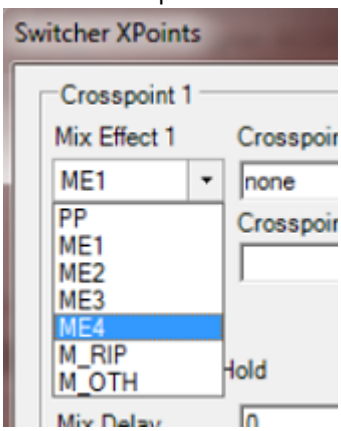
Creating the Wall Accessory Template

The wall accessory template is created as a standard template.

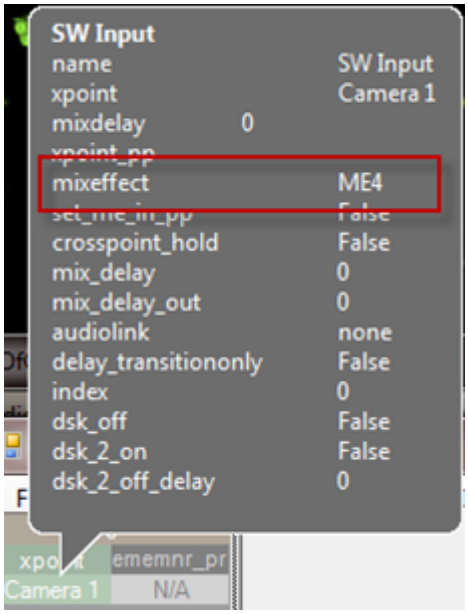
- You can create a new accessory template with a variant name of for instance "wall2".

Switcher Cross Point

- To change switcher crosspoints enable **Switcher crosspoint** in the Template Editor.
  The Switcher XPoints appear.
  Here you can set the ME-step you want for the wall taken template.
- When the template is set to the wall item it will use this ME step instead of the one specified in the template.



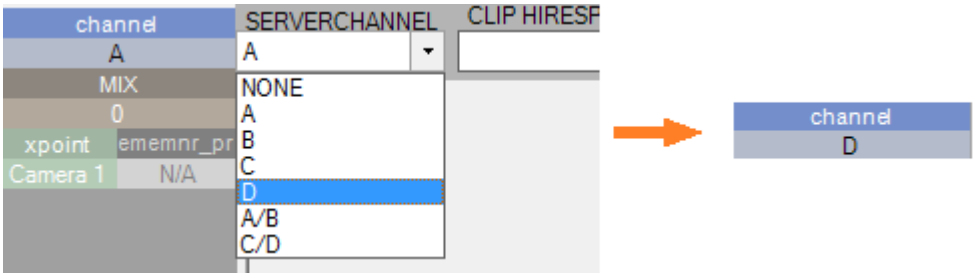
The ME-step will show as a mix effect in the Template Editor.



Video Server Port

If a video clip is wall taken; another video server should be used.

- This can be changed in the accessory.



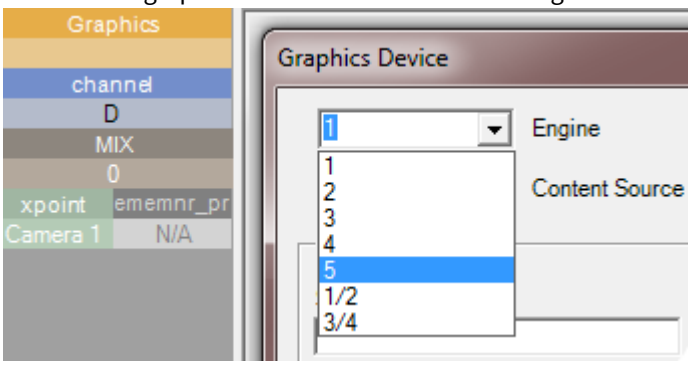
Make sure the video clips are available through this port.

AUX

- The AUX is controlled by the template but should be added to the accessory template.
  The setting will be replaced by the wall taken template.

Graphics

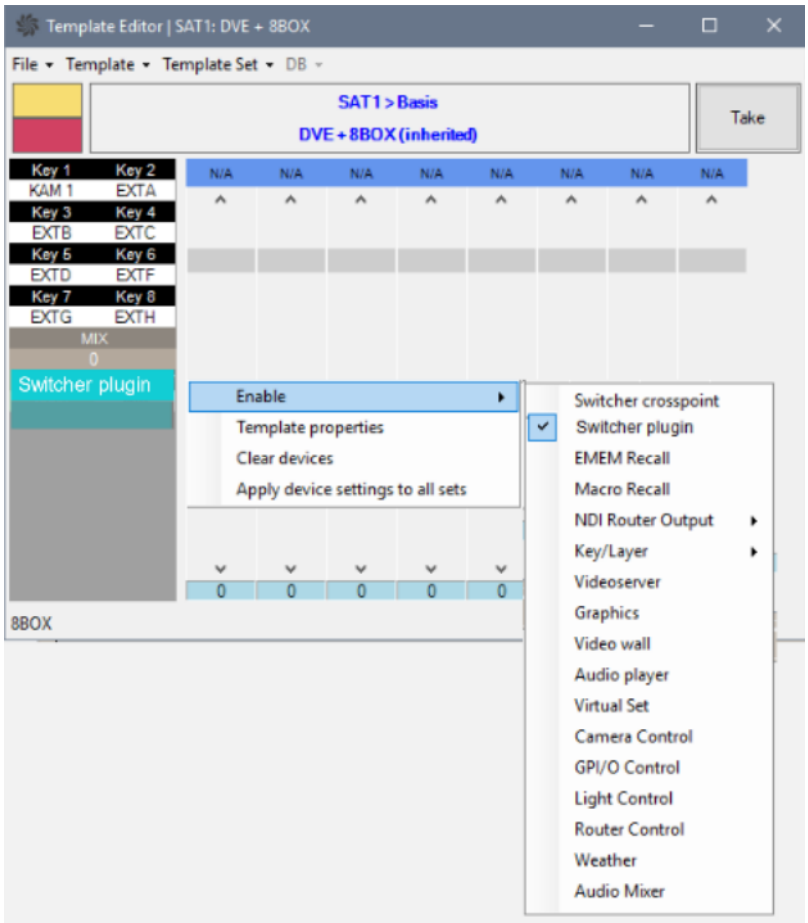
- To avoid conflict with other graphic elements, you may want the wall to run on its own engine. This can be changed in the accessory.
  A wall taken graphic element will then use this engine.



9.2.8 Switcher Plugin

These are advanced configurations to a standard switcher where functionality is extended. The example in this section refers to extending *Viz Engine* functionality.

A **Switcher plugin** option is available in the list of devices to be enabled on the template.



The **Switcher plugin** view introduces two new configurable features, *top scenes* and *presets*.

Static values can be specified directly on the template. Newsroom tags can also be configured, enabling control from within the NRCS.

Top Scenes

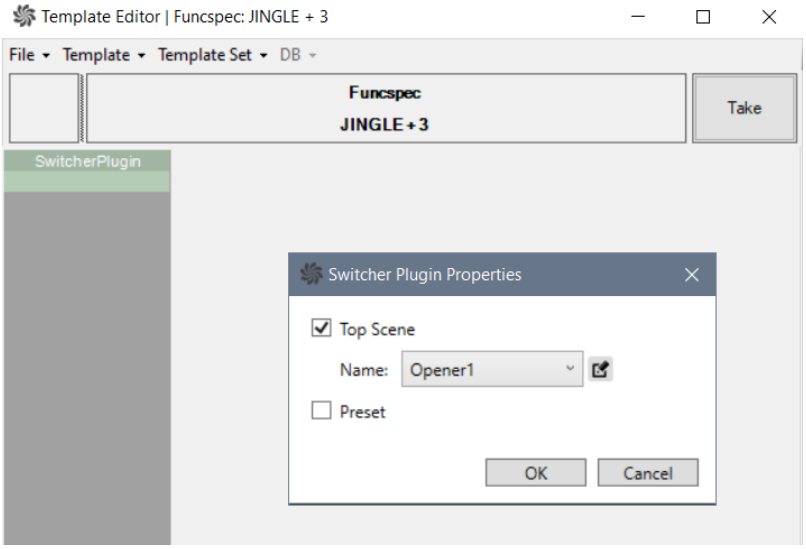
> **Note**: To work with top scenes, you must first perform the required setup, as described in the *Configuration* section of Viz Engine as a Switcher.

A typical use of top scenes is as an *opener*. For example, the top scene is played to introduce the 8 o'clock news show with many visual effects that fade to the following camera scene, and a news host taking over.

> **Tip**: For an in-depth explanation of Top Scenes and their time-related configurations, refer to the section Viz Engine as a Switcher (Top Scenes).

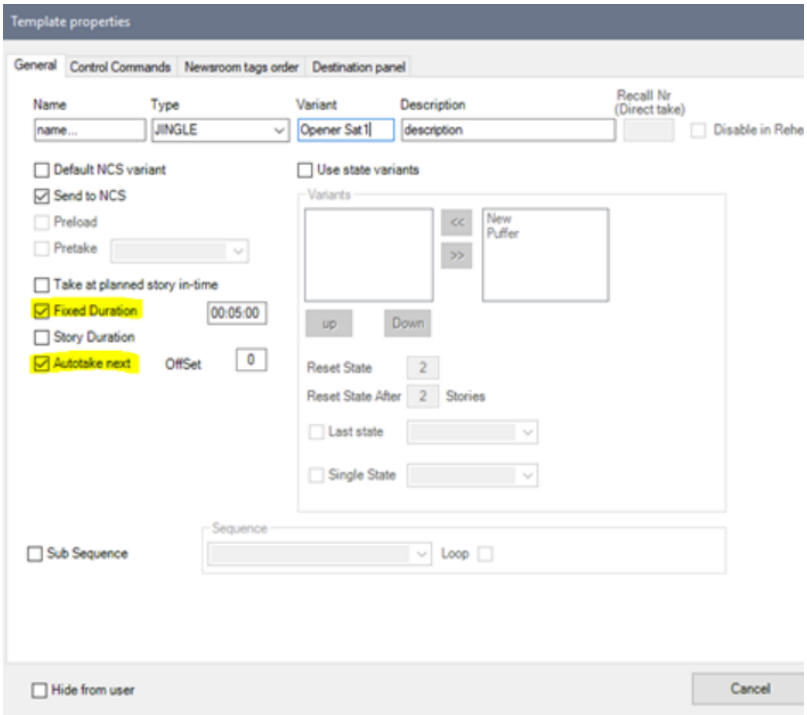
To create a template that plays a top scene

1. Create (typically) a Jingle template.
2. Enable **Switcher plugin** for the device.
3. Specify the top scene name from the list with the following options:
   a. None (empty).
   b. A list of top scene pre-configured in the A/V Setup.
   Select the required top scene (this is the name of the scene in the associated Graphics Hub folder).



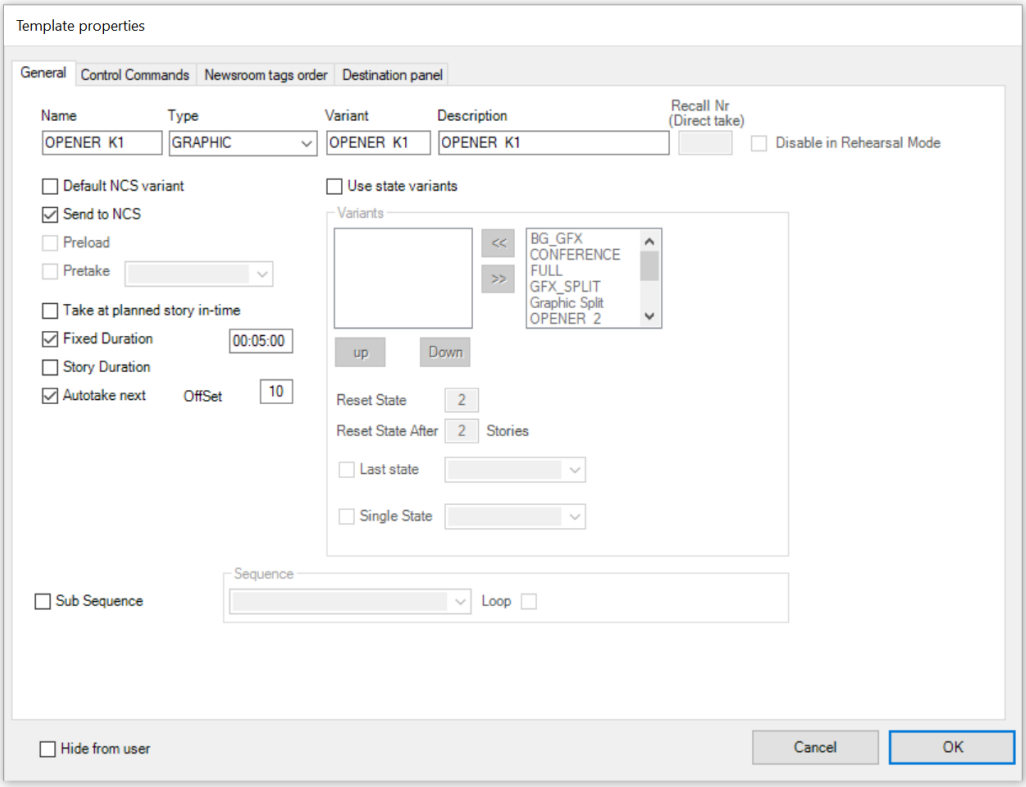
There are two time-related aspects when creating new templates for top scenes:

1. The time it takes for the scene to load. This is handled automatically by Viz Mosart when cueing the story item.
2. The time it takes for the scene to run. This is controlled with the **Fixed Duration** and **Autotake** configurations on the template.



Autotake

- You can add an *Auto Take* to the template to jump to next story item automatically (switch underneath the Top Scene)

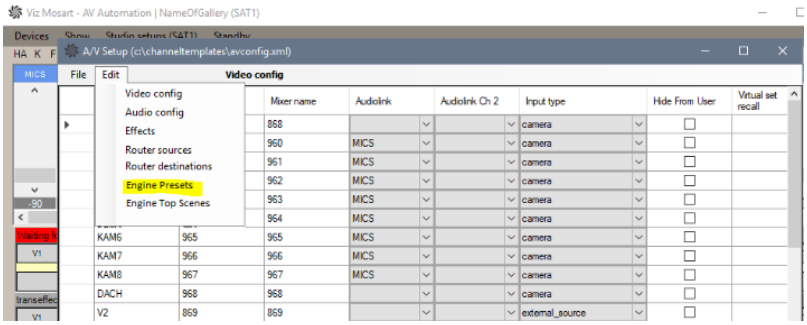


Presets

Select *Engine Presets* from **Edit** tab.

Use one of the following options:

- None (empty).
- A list of preset names pre-configured in A/V Setup (AvConfig.xml) file.

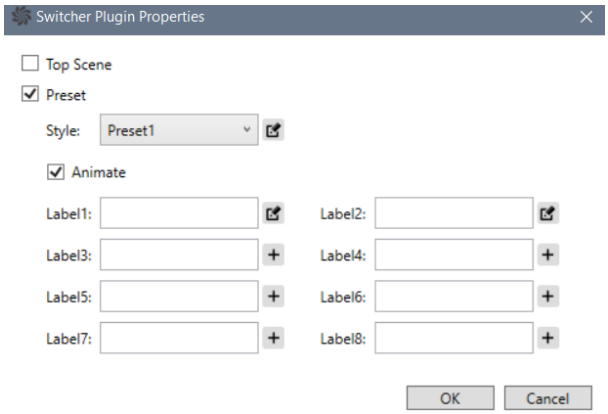


Transition Style

The transition style (hard-cut or animation), as well as several label values can be specified.

For example, on the DVE template above, where **Switcher plugin** for the device is available, the user can select the layout and looks of the DVE effect by specifying a preset style, the preset transition and the DVE window labels, as

required:

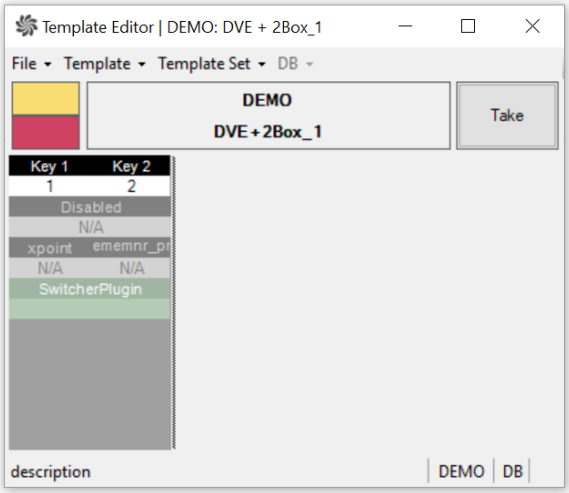


Switcher Plugin Additional Features

DVE Boxes (in a Preset)

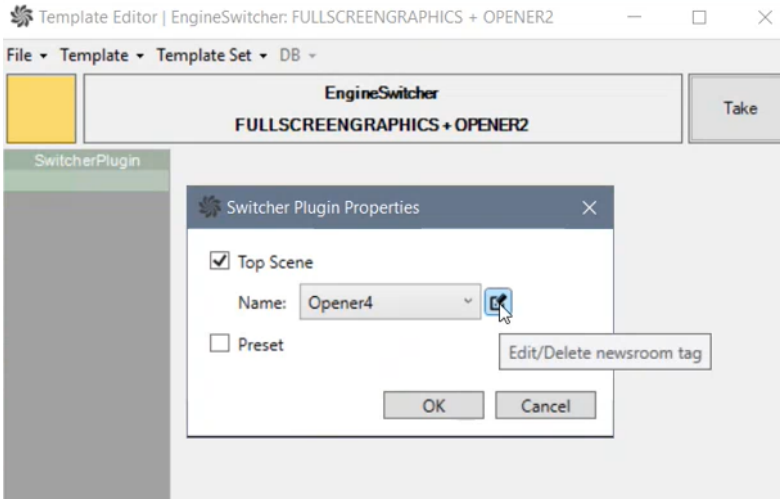
Simple 2-Box for Live sources

- Keyfill 1
- Keyfill 2
- *No* XPOINT !
- Switcherplugin



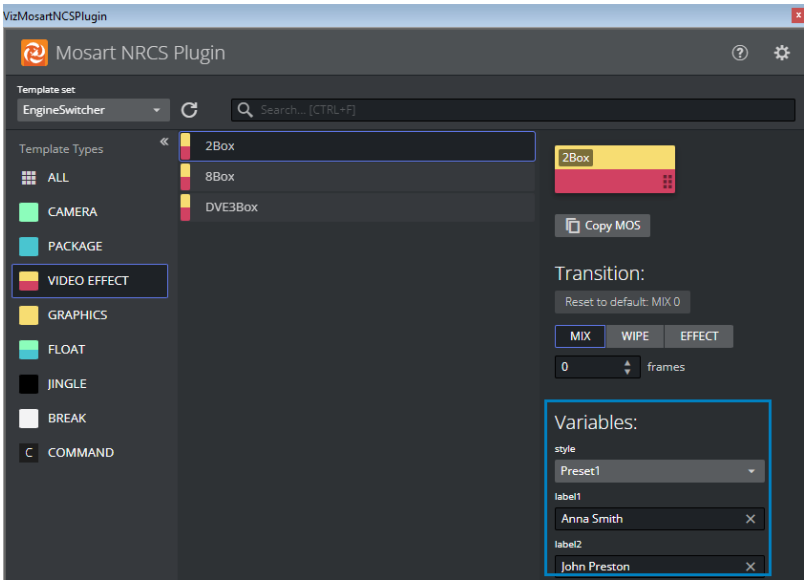
Switcher Plugin Settings

- Select the required Top Scene.
- Fill in any required labels.

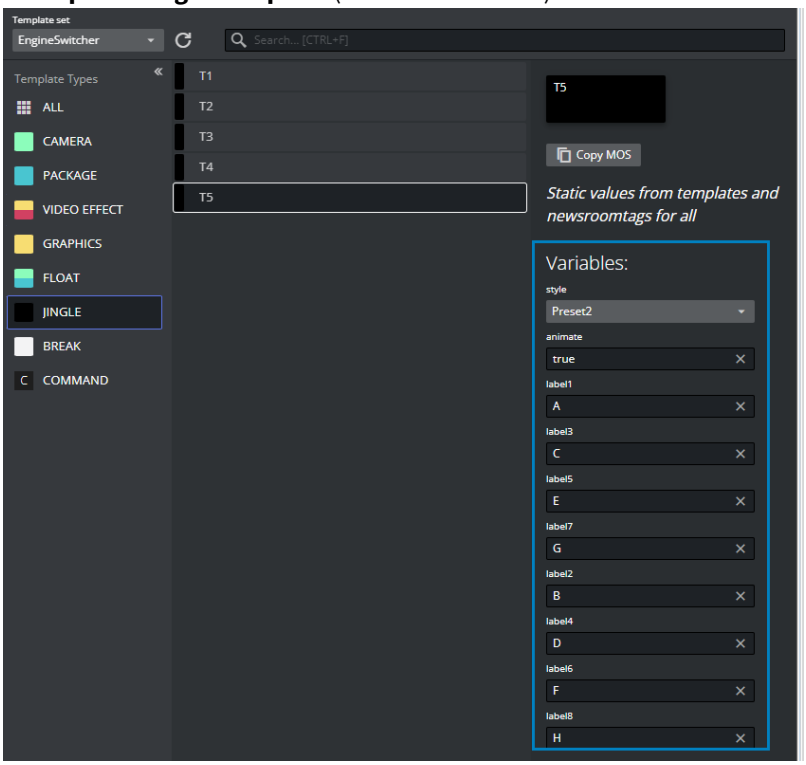


- Labels can be linked to an NRCS with a newsroom tag.
  Clicking the **Edit/Delete newsroom tag** icon, opens a mappings menu:

Example 1: 2-Box Template

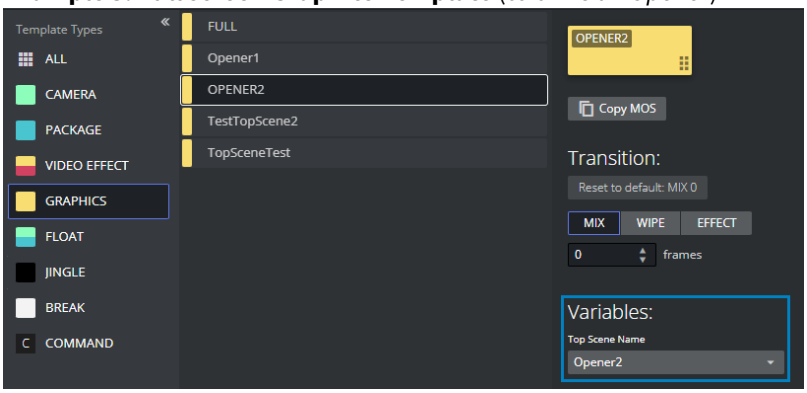


Example 2: Jingle Template (to control a *Preset*)



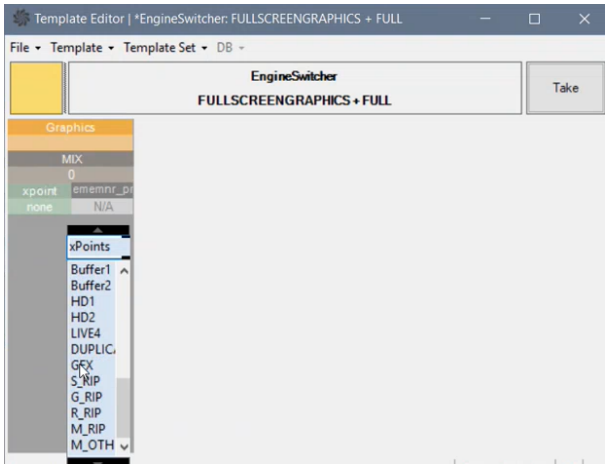
Note: To use the *animate* newsroom tag, set the **animate** dropdown option to *true*.

Example 3: Full screen Graphics Template (to drive an *Opener*)



Graphics

Static values are set in the Viz Engine, so always choose GFX for the switcher crosspoint.



See also: Viz Engine as a Switcher, section Template Design.

9.3 Working with Templates

This section contains the following topics:

- Template Operations
- Template Properties
- Template Device Functions
- Newsroom Tags
- Control Commands in Templates
- Destination Panel
- Empty Templates
- Configuring a REST-based Driver
- Mosart Templates Feedback to the NRCS

See also: Linking Device Properties and Newsroom Tags.

Copyright © 2026 Vizrt

9.3.1 Template Operations

- To Add a Template
- To Edit a Template
- To Remove a Template
- To Flatten a Template
- To Copy a Template

To Add a Template

You can add a new template to the currently selected template set.

1. Select **File > New template**.
2. Enter the new Template Properties.

To Edit a Template

- Select **Edit > Template properties**.

> **Note:** Template properties can also be accessed by right-clicking in the device's Function area. See the table Template Properties.

To Remove a Template

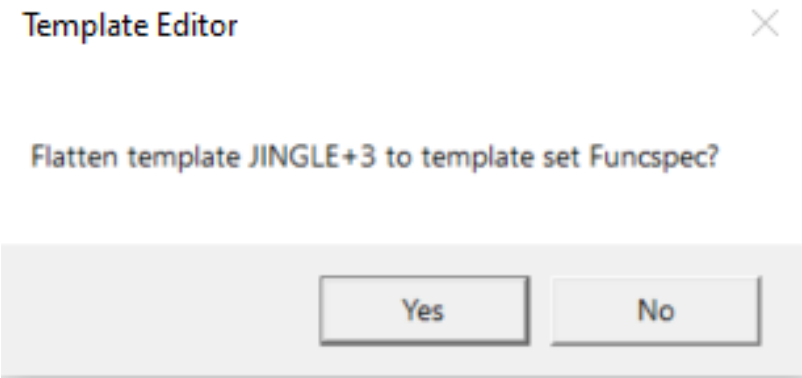
1. Select **File > Remove template**.
2. Confirm delete in the dialog box.

> **Note**: Removing a template from the base template set also deletes any overridden variant of the same template in any *inherited* (a different template set, hierarchically based upon the original template set) set .
> If this default behavior is *not* required (where we do *not* want to delete overridden templates when deleting a base template):
>
> 1. In AV Automation, open the Advanced settings by typing **Ctrl+Shift+S**.
> 2. Change the default setting of **DeleteOverriddenTemplate** to *False*.
> 3. Save and restart AV Automation.

To Flatten a Template

You can push a new template, created in another template set (that was created using the hierarchical approach), back to the originating template set.

1. Select **Template > Flatten**.
   A dialog box displays which template set the selected template will be flattened to.



2. Click **Yes**.

Note: For a description of Hierarchical templates, please refer to Working with Template Sets.

To Copy a Template

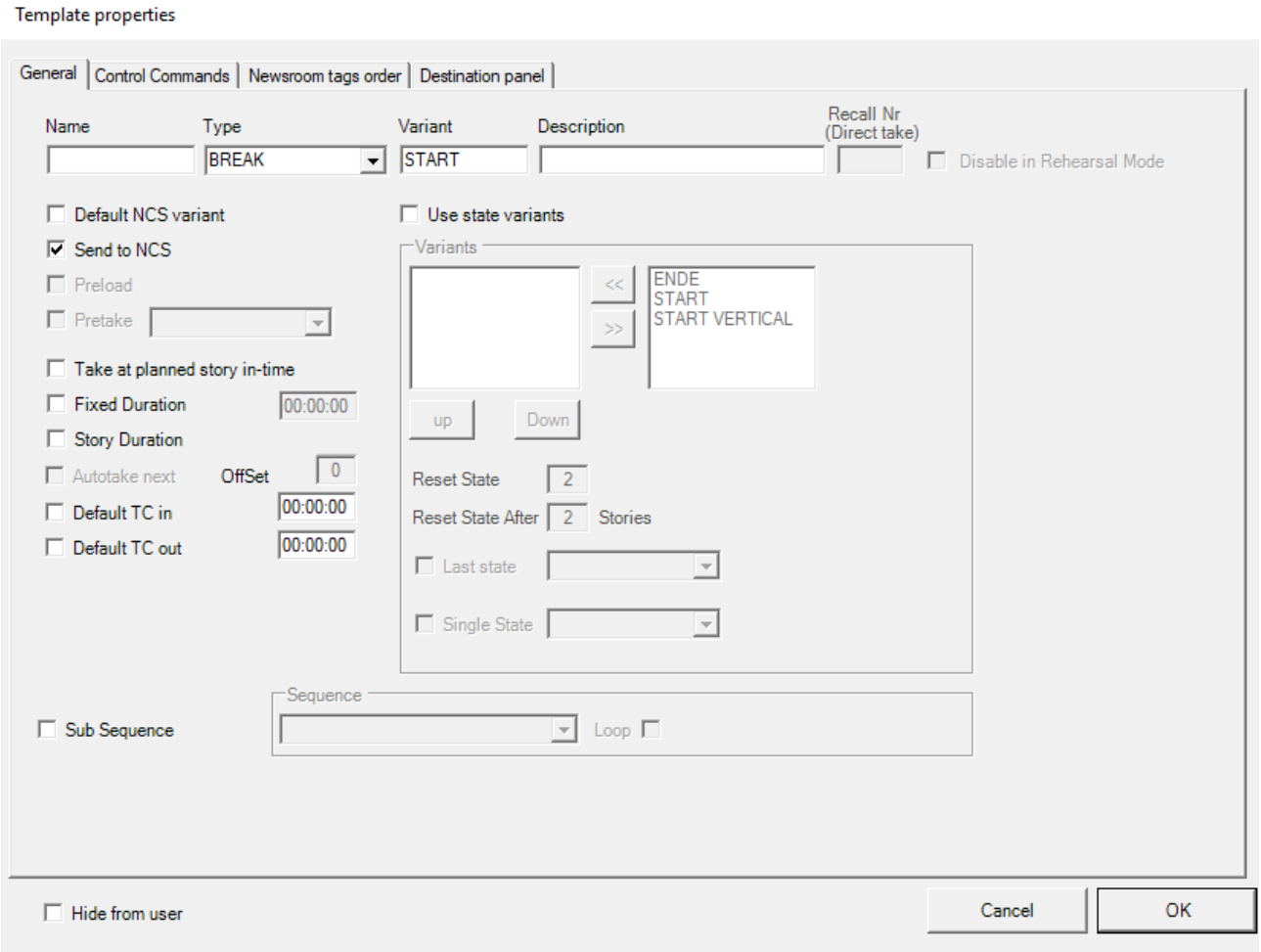
You can copy a template from one template set to another template set.

1. Open the template you want to copy.
2. Select **Edit > Copy template**.
3. Switch to the other template set, and then apply **Edit > Paste template**.
4. Select **Edit > Paste to all** to copy the template to all template sets.

9.3.2 Template Properties

This section explains the configurable fields on the **General** tab of the **Template Properties** menu, and also clarifies the function of a *Sequence*.

- Configuration Descriptions
- AutoTake Timings
  - Autotake next Templates
  - Duration
- Sequences
  - Recalling a Sequence from Template
  - Recall a Sequence from a Direct Take Template
  - Controlling a Sequence



Configuration Descriptions

Item	Description
Name	The name of the template. This value will only show in log files. English language is preferred, for making it easier to understand log-files.
Type	The different primary and secondary templates types are described in the *Viz Mosart User Guide,* section *Viz Mosart User Interface* under *Rundown Window*.
Variant	The variant is a unique name for the selected type, and should match the value entered in the newsroom system.
Description	Description of the template (optional). English language is recommended, for future maintenance purposes.
Recall Nr (Direct take)	(Applies only for templates in the Directtakes template set). One or more decimal digits. If this direct take template is to be recalled from the GUI using the numeric keypad, a maximum of four digits is allowed. (If more than four digits are used, the template may still be executed using e.g. a control command.) The number must be *unique* among all direct take templates. **Note:** Avoid starting numbers with zero (0). For example, use 56 instead of 0056.
Disable in Rehearsal mode	(Applies only for templates in the Directtakes template set). The direct take will not be executed when in Rehearsal Mode
Default TC in	
Default TC out	
Default NCS variant	This sets the template as the default variant for the selected type.

Send to NCS	When selected, allow Manus Administrator to send this template to the NRCS, using MOS NRCS feedback.
Preload	When selected, enable the preview functionality for the Accessory template. The Accessory template must have time code 00:00 for this to function. *Only valid for Accessory templates.*
Take at planned story in-time	If you want a story to start at a specific time of day, you can add a template with this option checked, to a story that has a set "hit time" in the NRCS. The "hit time" must be included in the story data sent to Viz Mosart, and the field-name must be mapped to **back_time** in Newsroomsettings. See Story External Metadata.
Fixed Duration	Make the template always have the same length (overriding the time from the NCS) in the Viz Mosart rundown. Select this option and enter the time in mm:ss:ff (minutes:seconds:frames). This can also be combined with Autotake to make a frame accurate continue to the next story. Remember to write the whole number each time, you cannot edit a single character at a time.
Story duration	When selected, the duration of the template will be set equal to "story editorial duration", coming from the NRCS. The "story editorial duration" must be included in the story data sent to Viz Mosart, and the field-name must be mapped to **story_duration** in Newsroomsettings. See Story External Metadata. If more templates are added to the story, their duration will be in addition to the "Story duration" template, so calculated total duration of templates will be greater than "story editorial duration".
Autotake next	(Available for *primary* templates only, cf. **Type** above.) When selected, automatically perform a **Take Next** (**F12**) at the end of the template's duration, with an offset as described in **Offset** below

Offset	Set the offset when Autotake is enabled to adjust the Viz Mosart continue action at the end of the event. The value is in frames. A positive value will make the continue action happen later. A negative value will make it happen earlier.
Sub Sequence	See separate explanation below this table.
Sequence	See separate explanation below this table.
Loop	See separate explanation below this table.
Hide from user	When selected, this template is excluded from the template list in both the Mosart UI's **Quick Editor** and the **NRCS Plugin**, making it unavailable for user selection.
Use state variants	Please refer to Other Template Functionality for explanation of state variants.
Control Commands (tab)	If the template should have *control commands* attached, configure here. Please refer to Control Commands in Templates for further instructions.

AutoTake Timings

An item has a *duration*. The method of determining this duration depends on the template type. For example, for a *Package*, it is the duration of the *video clip* that determines duration.

Duration can also be set from the NCS.

Autotake next Templates

An **Autotake next** template enforces that the *next* item will be taken automatically when *this* item is 'finished'.

For most Autotake next items, the *next* item will be taken when the *duration*, as defined below, has elapsed. An exception is an item with *clips, where Mix Delay* will also have to elapse before the next item is taken

Duration

- For an item based on an **Autotake next** template, the Offset (*positive, 0* , or *negative*) is *added* to the duration.
- If the underlying template has an *effect* transition, the *duration* of the effect (as defined in the Effects table in A/V Setup) is *added* to the duration.

Sequences

A sequence is a saved story (including primary and secondary items with relative timing information).

Sequences are created in Viz Mosart GUI.
This is described in the *Viz Mosart User Guide*, under Operation, in section Creating Sequences.

Recalling a Sequence from Template

To recall a sequence from a template

1. Select **Sub Sequence.**
2. Select the Sequence to recall. (The available options are the sequence names saved to the template set). For a direct take template, *all* sequence names, regardless of template set, are eligible, see Recall a sequence from a direct take template.
3. If required, select **Loop**.

When a sequence is recalled from a template, actually only the *secondary* items of the *first* primary item are recalled (but still with the same timing information that it had in the saved story sequence).

Recall a Sequence from a Direct Take Template

To recall a sequence from a direct take template

- Sequences may be recalled from direct take templates just as from other templates.
  However, when taking a direct take template, a sequence will be recalled only if the *current* template set (*at that time*) has a sequence with the same name chosen. (And then that sequence will be recalled.)
- Any take out logic might not work as expected if the direct take is not fired at the beginning of the current main template.
- When using direct takes with sequences it is recommended to recall them *from a primary template* and *not* use them with keyboard shortcuts.

Controlling a Sequence

When a sequence has been started it can be controlled using keyboard shortcuts with Control Command - SEQUENCE.

See the table *Command Values and Parameters*, COMMAND SEQUENCE, Value LOOP, STOP_LOOP, STOP, START.

9.3.3 Template Device Functions

This section lists a selection of device properties:

- Working with Device Functions
- Testing a Template
- Video Switcher Crosspoint
- Video Switcher Transition
- Video Switcher Register/Timeline Recall
- Macro Recall
- Video Switcher Auxiliary Bus Delegation
- Keyfill: Video Switcher Key Bus Delegation
- Video Server
- Graphics
- Video Wall Register Recall
- Audio Player
- Virtual Set
- Robotic Camera Control
- GPO
- Light Control
- Router Control
- Audio Faders
- Generic REST commands

Working with Device Functions

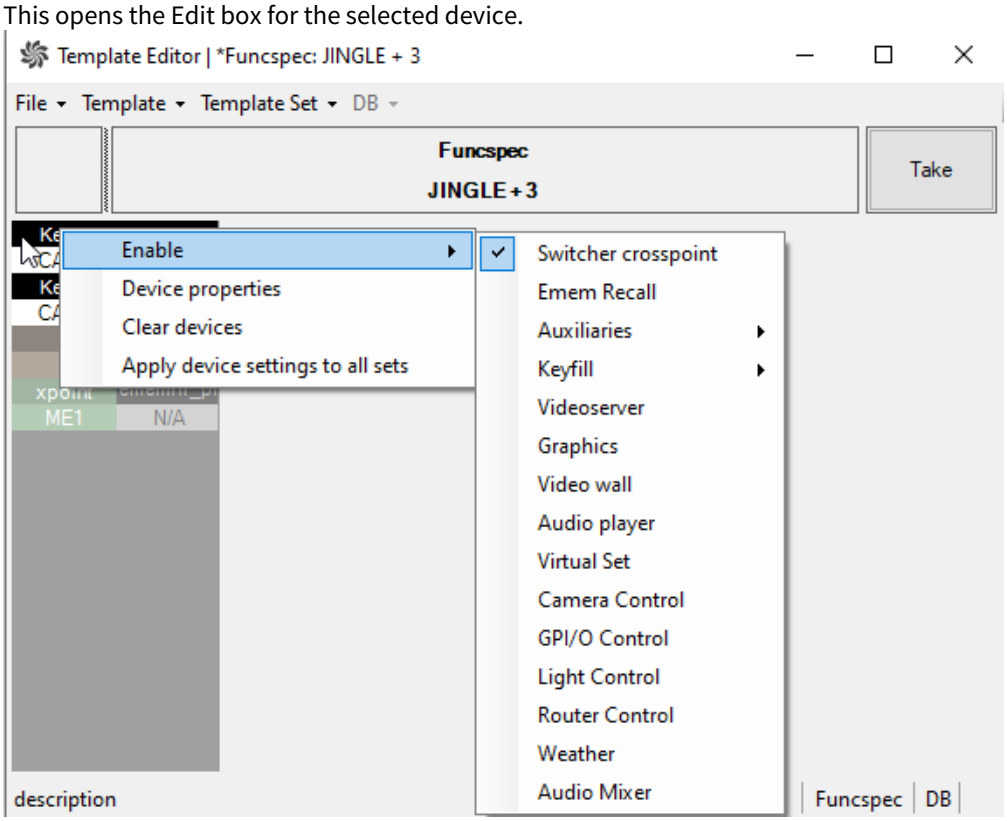
Right-click inside the device function area to open a menu containing the following options:

- **Enable**: These commands are described in more detail below.
- **Template properties:** Opens the current template's properties page, see Template Properties in section *Working with Templates*.
- **Clear devices:** Removes all devices from the device function menu, after confirming in a pop-up message window.
- **Apply device settings to all sets:** Applies all the device settings to all templates of same type and variant throughout all template sets.
- **Mark as intentionally empty:** This option is available for *empty templates* only and shows up when the **Use 'Mark template as empty' feature** is enabled in Settings Editor MOS-General. See Empty Templates for more details.

To enable device functions

1. Right-click in the device's Function area (black row).

2. Select the device to configure from the **Enable** menu.
   This opens the Edit box for the selected device.



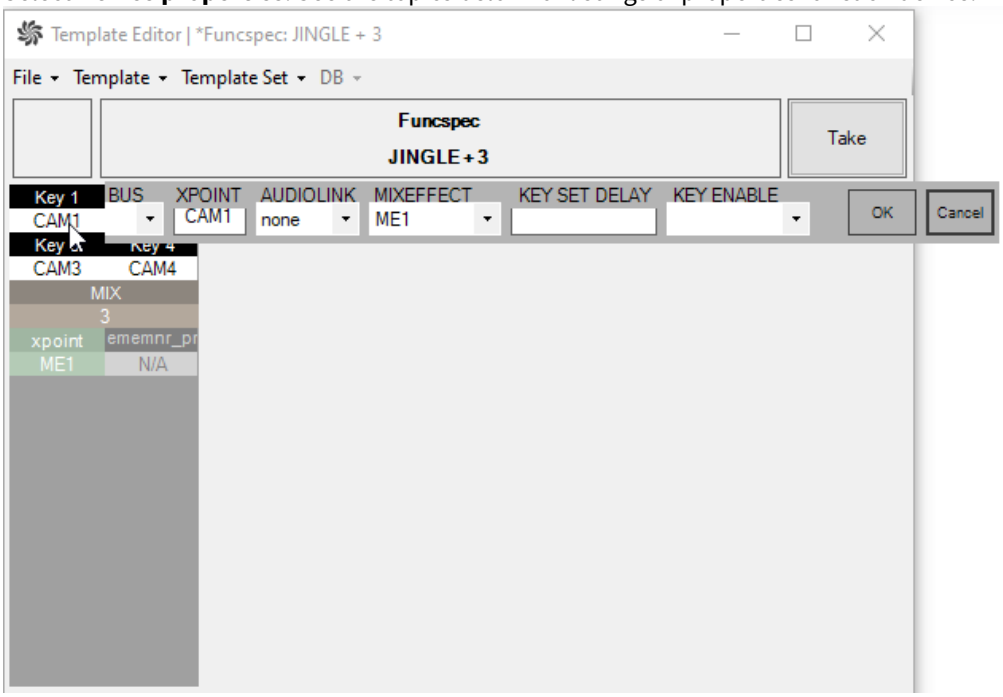
Info: Depending on the protocol chosen, the caption **EMEM Recall** may alternatively be:

- **Emem Recall**
- **Dmem Recall**
- **Snapshot Recall**
- **Memory Recall**

To modify device functions

1. Right-click in the device's Function area (black row).

2. Select **Device properties**. See the topics below for listings of properties for each device.



Info: Depending on the Protocol chosen, the caption **Macro Recall** may alternatively be:

- **Custom Control**

To remove device functions

1. Right click in the Device function area
2. De-select it from the **Enable** menu.

Linking Device Properties and Newsroom Tags

To make a property editable from the NRCS

- Right-click over **Property** and select **Add newsroom tag**.
  The property is then reassigned to the value supplied from the newsroom system.
- In the grid-based device editors — Robotic Camera Control, Router Control and GPO — right-click over a cell instead, and select **Add newsroom tag** to make that column's attribute editable from the NRCS.

If several device properties are given to a newsroom tag with the *same* name in a template, all devices use the value given in the NRCS.

See also: Newsroom Tags.

Testing a Template

You can test the behavior of your template by simulating a Preview (cue) followed by a Take.

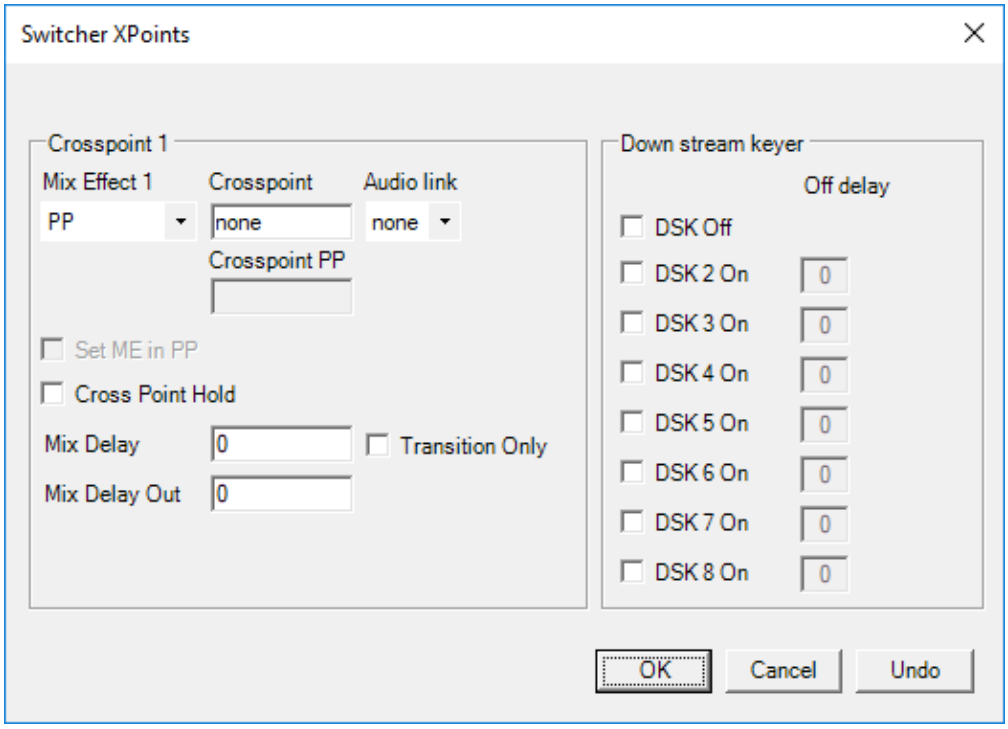
To test a template

- Press the **Take** button to the right of the template heading.

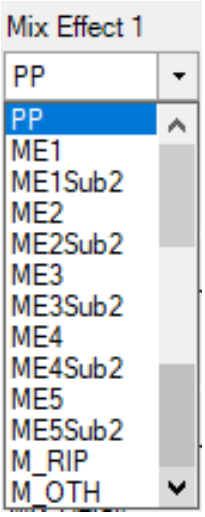
This cues the template in Preview, and then five seconds later takes it to Program with the selected transition.

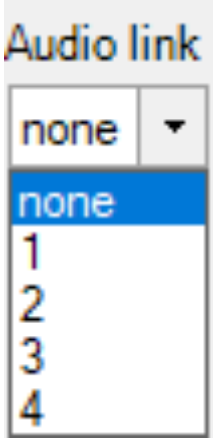
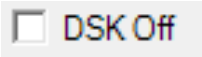
Video Switcher Crosspoint

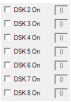
The video switcher crosspoint enables switching on the video switcher. The A or B bus assignment of the mixer effect (PP/MEx) is automatically handled by the automation.



Guidelines for Video Switcher Crosspoint Configuration

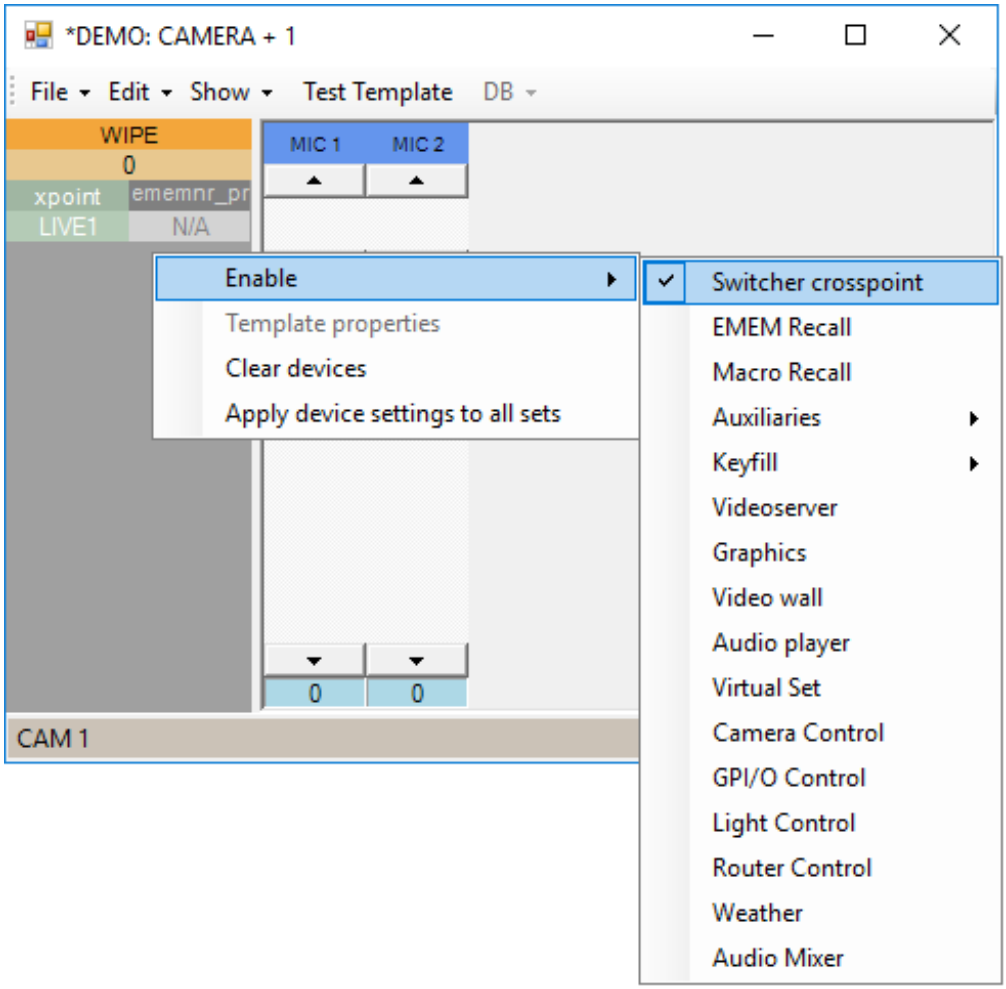
Item	Description
Mix Effect	Selects which Mix Effect (ME) bank on your vision mixer that you are currently handling. **Note** : Newsroom tag available.  **Info:** For vMix, PP maps to the main vMix output. ME1 maps to the *first* Mix input, ME2 to the *second*, and so on. **Info:** For ATEM, both PP and ME1 map to the *same* physical ME1 on the switcher. The difference is only relevant for keyfills. See *Guidelines for Keyfill Configuration* below.
Crosspoint	Crosspoint on the video switcher. Default value: none • Enter a text value (shall be same as in the list of available x-points for Video configurations) or a predefined value like M_RIP or M_OTH used for the ME dynamic allocation and S_RIP for dynamic allocation of clip sources, often referred to as an A/B roll setup. **Info:** Newsroom tag available.

Item	Description
Audio link	Default value: *none* 
Crosspoint PP	If another Mix Effect (ME) than *PP* is selected, then you can select a crosspoint for program by writing the crosspoint name in the field.
Set ME in PP	If another Mix Effect (ME) than *PP* is selected, check this box to make that ME the crosspoint for PP. (Checking this box disables XPOINT PP).
Cross Point Hold	Keeps the crosspoint on the selected **Mix Effect**.
Mix Delay	Offset, in frames, of when to perform the transition.
Mix Delay Out	Delay, in frames, of when to perform the next transition.
Transition Only	Only delay the switcher transition (sub items will perform).
DSK Off	Turns off the Downstream Keyer (DSK) on the video switcher when the template is active in program. The keyer selected as DSK in the switcher device property **(Devices** > **Properties** > **Vision Mixer >** ***selected DSKNr***), will be used. 

Item	Description
DSK 2 On, ..., **DSK 8 On**	Downstream Keyer (DSK) 2-8 on. Turns on the DSK 2-8 (PP key 2-8) on the video switcher when the template is active in program. 
Off delay	Delay in frames to set the DSK 2-8 off.

Video Switcher Transition

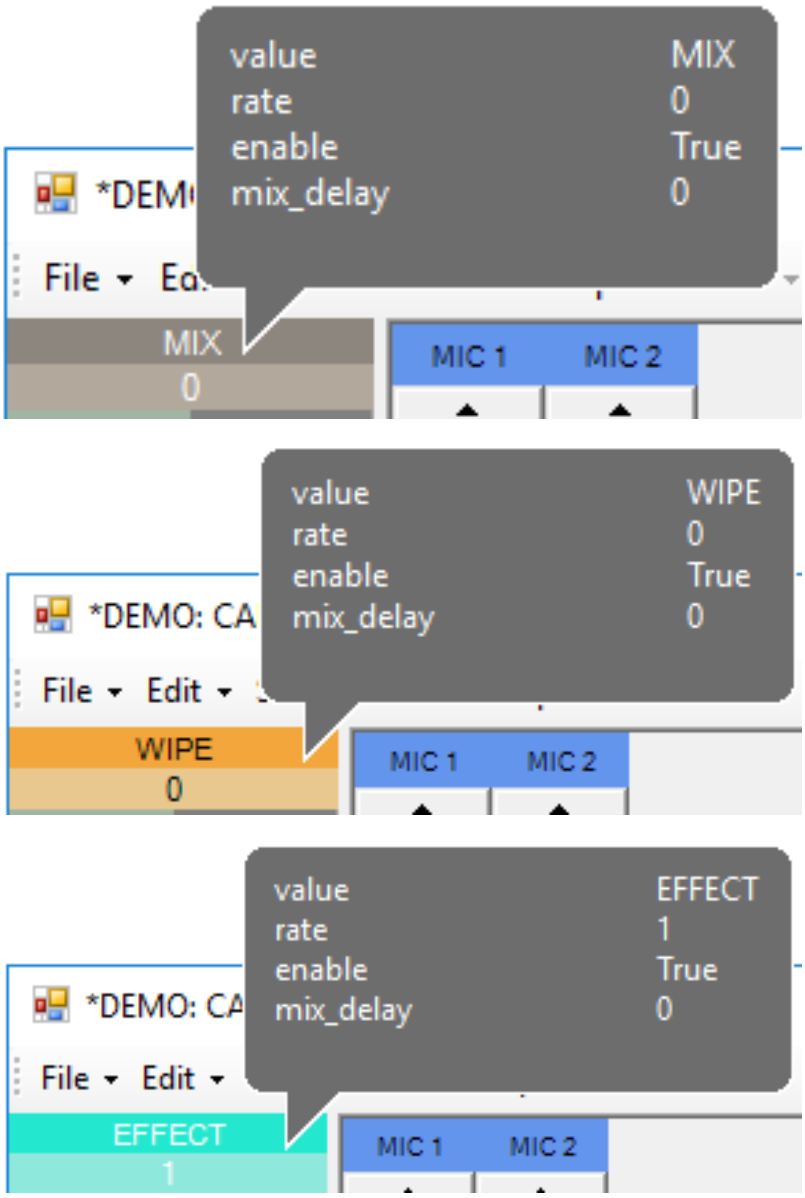
The transition device is only available when video **Switcher crosspoint** is enabled.



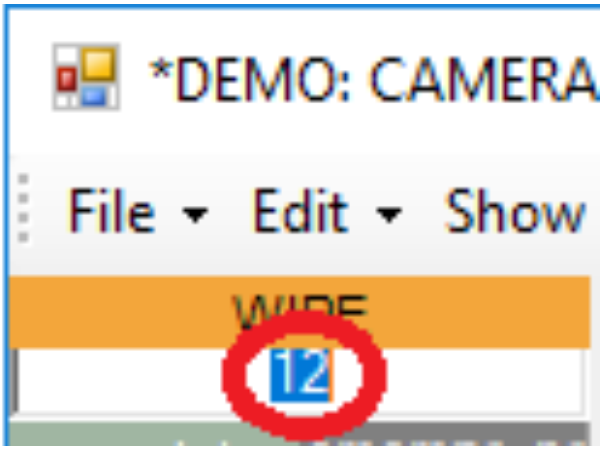
To define video switcher transitions

This procedure is based on the example in the screenshot.

1. Click on **MIX/WIPE/EFFECT** to cycle through these three transition types.



2. Click directly on the selected value (in the screenshot above, this is either *0* or *1*).
3. Enter
   a. The default duration (in frames) for the Mix/Wipe transition, or
   b. The effect number for the effect transition.



The transition is performed with the auto-transition functionality in the video switcher.

Usage and Tips

- **To add a cut**: Use a MIX transition with `0` frames duration.
- **To add a mix delay**:
  Right-click on the device (**MIX/WIPE/EFFECT**), select the number (below **Disable**), and enter another number (in frames).
  Adding a mix delay delays the start of the transition, compared to other device commands, when the template goes On Air.
- **To disable a transition**:
  Right-click over the device and select **Disable**.

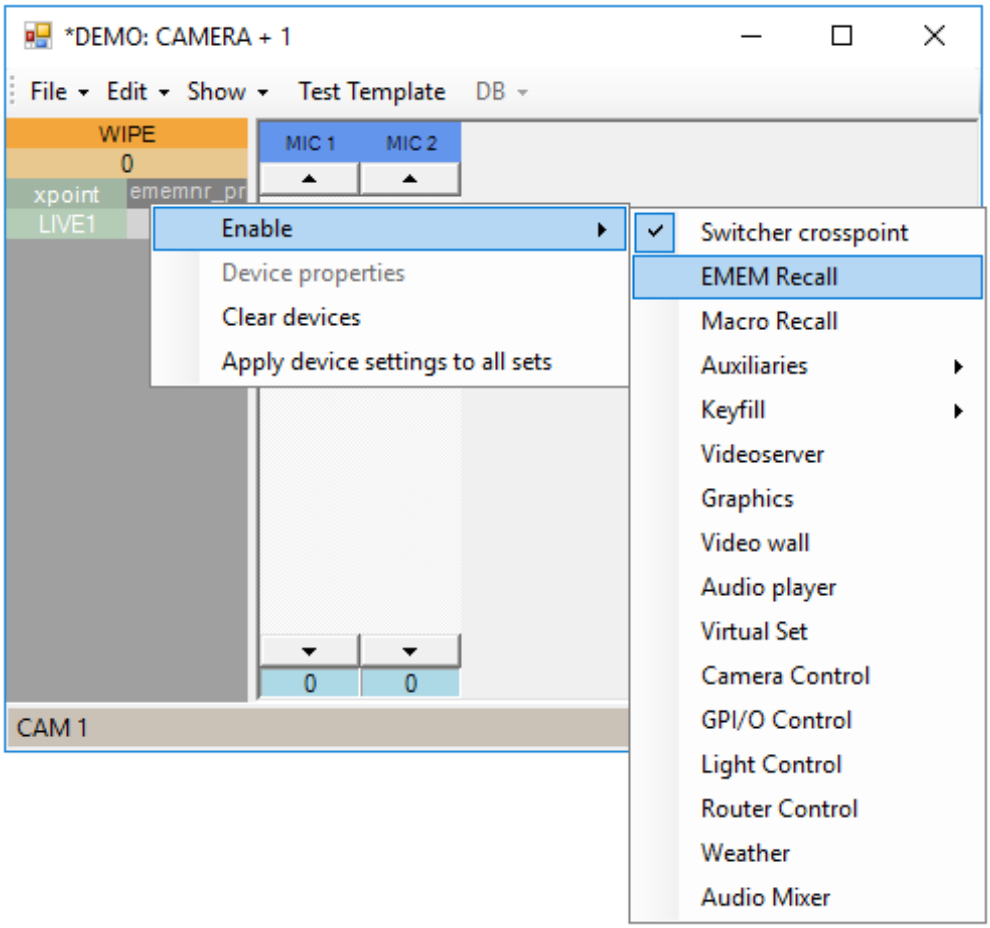
Video Switcher Register/Timeline Recall

Video Switcher Register/Timeline Recall (**EMEM Recall/Emem Recall/Dmem Recall/Snapshot Recall/ Memory Recall)** will recall a register and/or timeline in the video switcher.

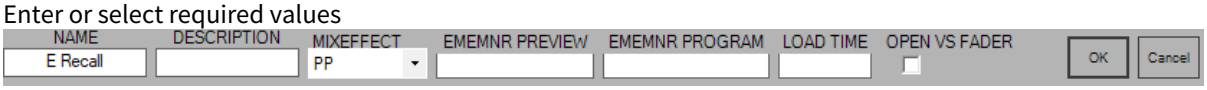
To open the video switcher Register/Timeline recall

This procedure is based on the example in the screenshot.

1. Open **Template Editor**.
2. Right-click over **ememnr_pr**
3. Select **Enable > EMEM Recall**



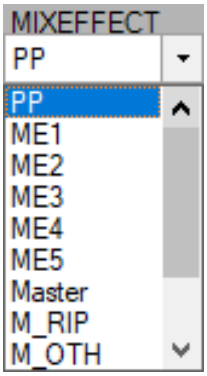
4. Enter or select required values



Info: If a Register recall is paired with the video switcher key bus delegation and/or auxiliary bus delegation, the *key/aux bus delegation* should be *disabled in the Video Switcher Register* to prevent a conflict between the stored delegation and the Viz Mosart-assigned delegation.

Guidelines for Video Switcher Register/Timeline Recall Configuration

Item	Description
NAME	Name of Register/Timeline.
DESCRIPTION	Description of Register/Timeline.

Item	Description
MIXEFFECT	Select the **MIXEFFECT** (ME) from where the Register/Timeline should be recalled. Drop-down menu values are *PP* (Program), *ME1*, *ME2*, *ME3*, *ME4*, and *Master*. Default value: *Master.* (Depending on the Protocol chosen, there may be additional values).  To dynamically choose *ME1* or *ME2*, values *M_RIP* and *M_OTH* are available. **Info:** Newsroom tag available
EMEMNR PREVIEW	Register/Timeline on the video switcher that will be recalled when the template is cued in Preview.
EMEMNR PROGRAM	Register/Timeline on the video switcher that will be recalled when the template is taken to Program.
LOAD TIME	Delay time (in frames) between when the mixer register is recalled (in Program or Preview) and when all other template instructions are executed.
OPEN VS FADER	When selected, Viz Mosart opens the fader specified in **Audio effect server**, defined in the **Audio** tab of **Device Properties** in AV Automation.

Macro Recall

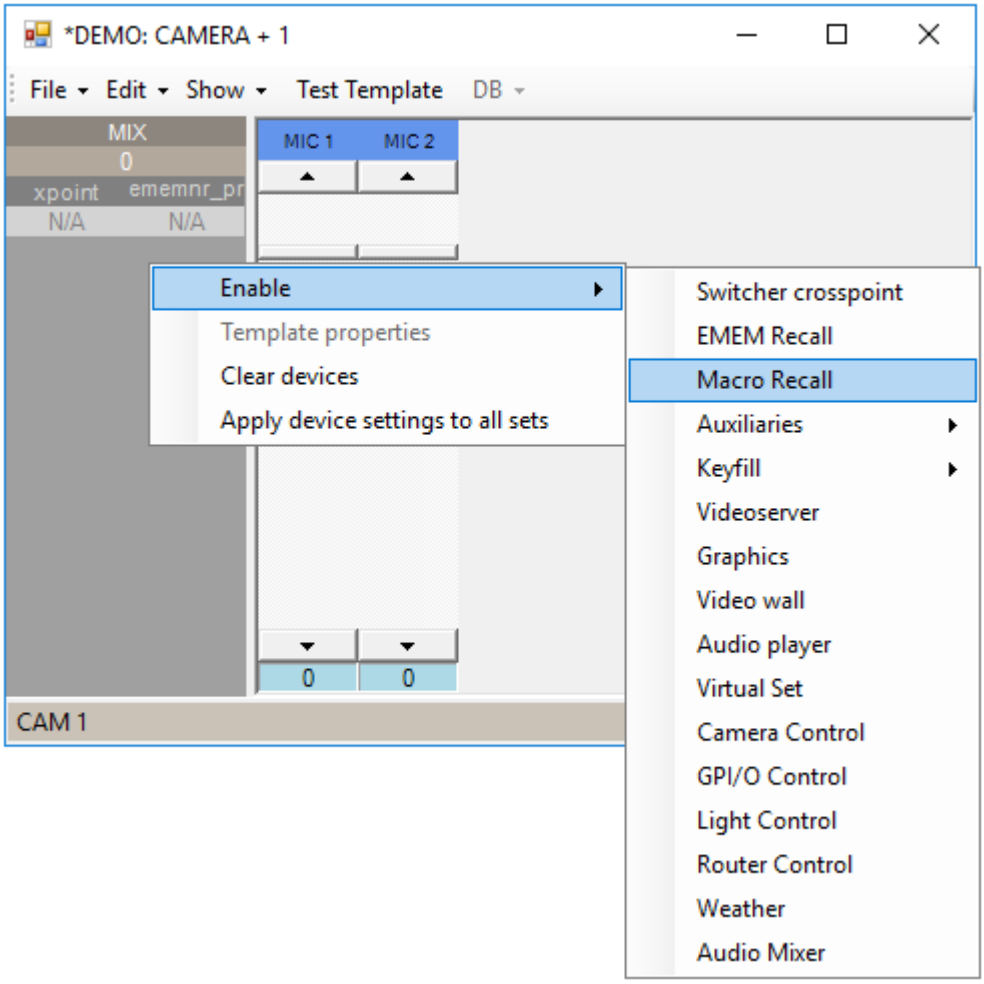
Macro Recall is available with the following vision mixer protocols (selected in the configuration menu AV Automation Devices Vision Mixer):

- Ross Acuity
- Ross Carbonite
- GV CPL
- GV KAHUNA (Kahuna/Kula)
- SONY SERIAL TALLY
- TriCaster
- ATEM

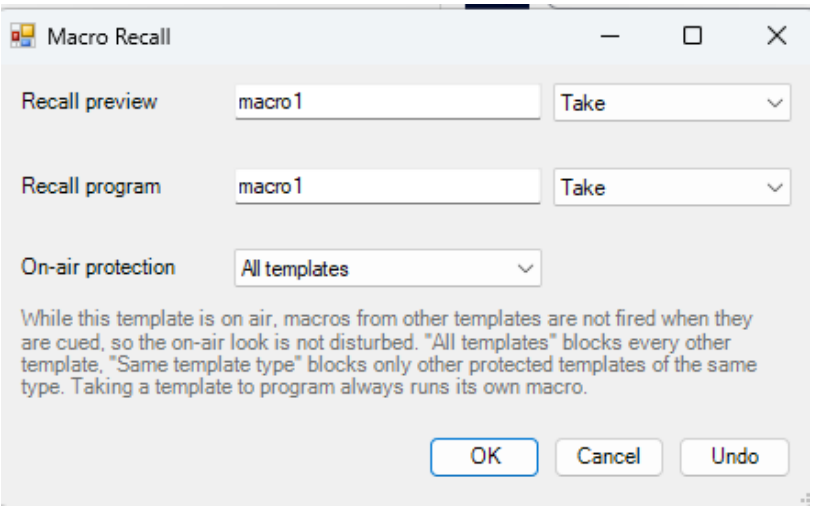
- vMix

To open a Macro Recall

1. Open **Template Editor**.
2. Navigate to **Enable > Macro Recall**



3. Enter or select required values.



Guidelines for Macro Recall Configuration

Item	General description	Drop-down	Text field
Recall preview	Action to be taken when entering preview mode.	See below	See below
Recall program	Action to be taken when entering program mode.	See below	See below
On-air protection	Prevents macros in other templates from changing the on-air look while this template is on air.	See below	N/A

Drop-down Menu Alternatives

Drop-down option	Description
None	No action.
Recall	Macro is prepared (or cued).
Take	Macro is taken.
Recall and Take	Macro is prepared and taken.
Load project	Project is loaded. Supported only by Kahuna/Kula and vMix. **Warning: Load Project** could affect the performance of the switcher. Use with caution during a running show.

Text field format depends on the mixer chosen

Mixer	Text field	Remark
ATEM	Name or index (starting from 1) of the macro.	**Load project** not supported.
Acuity	One bank character and two macro characters. Example: 102 (recall custom control 02 on bank 1)	**Recall** (and Recall and take) only. Take does nothing. Note that Device is named *Custom Control* instead of *Macro Recall*.
Carbonite	One bank character and two macro characters. Example: 215 (recall custom control 15 on bank 2)	**Recall** (and Recall and take) only. **Take** does nothing. Note that Device is named **Custom Control** instead of **Macro Recall**.
GV CPL	Macro number (integer) in the range *1-999*.	**Recall** (and Recall and take) only. **Take** just takes macro already recalled.
Kahuna/ Kula	The meaning and format of the text fields depend on the selected drop-down option: **Drop-down option** / **Description** Recall: Two-digit project number and three-digit macro number in the format *pp.mmm*. Take: As for Recall above, or three-digit macro number (*mmm*). Recall and Take: As for Take above (if *mmm* only, the Recall is omitted). Load project: Project number in the range *0-100* (*100* meaning no project).	
SONY SERIAL TALLY	Macro number (integer) in the range *1- 250*.	Recall (and Recall and take) only. Take just takes macro already recalled.
TriCaster	Plays a macro having the name provided or the unique ID (GUID).	Take only

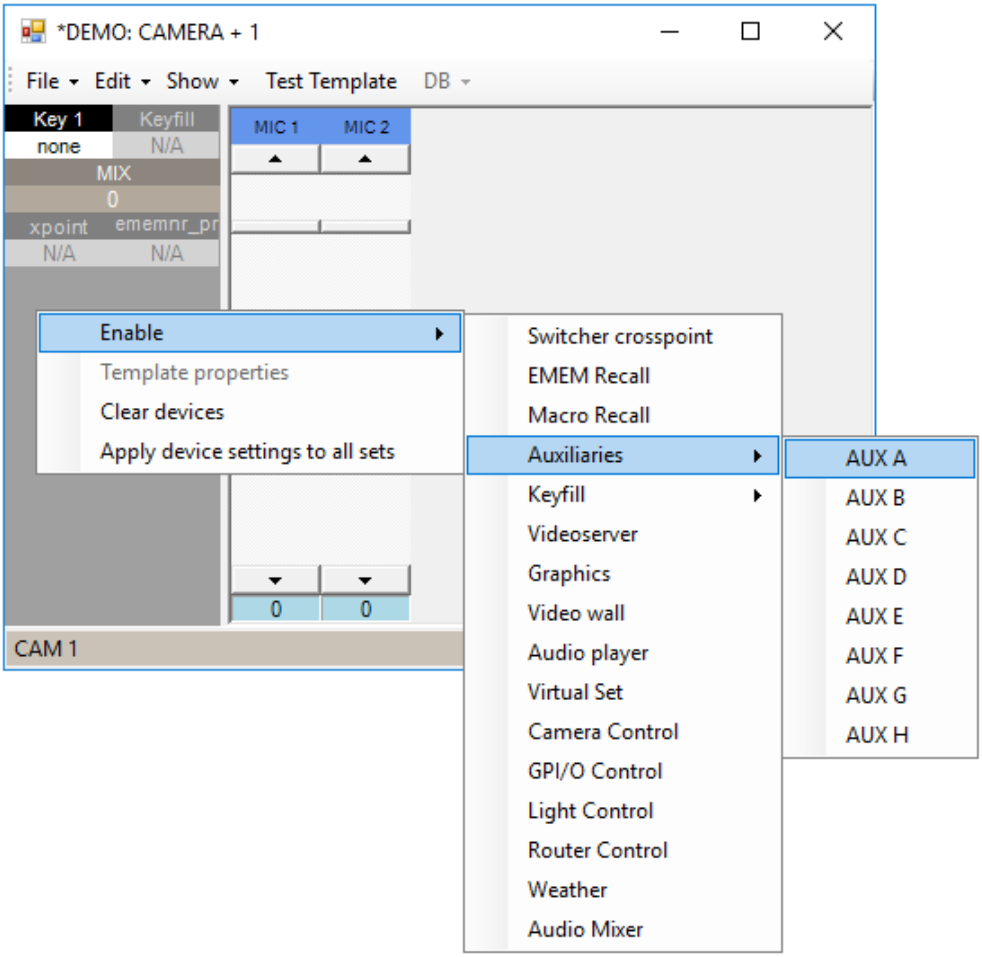
Mixer	Text field	Remark
vMix	Name of vMix script to execute. **Load project:** Path to preset file to load.	**Load project:** Absolute or relative path from the vMix installation folder (the `.vmix` extension is optional).

Drop-down option	Description
Off	No protection. Macros always run as configured. This is the default.
All templates	While this template is on air, a macro in any other template is not fired when that template is cued.
Same template type	While this template is on air, a macro is not fired when a template of the same type, for example another DVE, is cued and also has on-air protection enabled. Templates of other types are not affected.

Video Switcher Auxiliary Bus Delegation

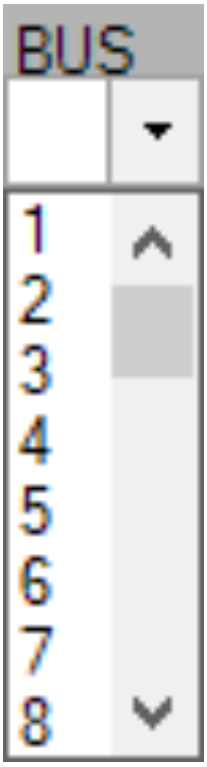
Auxiliary bus delegation enables routing of internal video switcher signals to the auxiliary outputs of your video switcher.

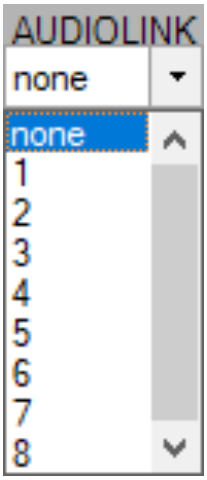
> **Info**: TriCaster video switcher uses *NDI outputs*, not auxiliaries. On the template, the **Auxiliaries** device is instead displayed as *NDI Router Output*.



- Type or select relevant values.

Guidelines for Auxiliary Bus Delegation Configuration

Item	Description
BUS	Selects the auxiliary to assign a new source.  **Info:** For vMix, bus 1 corresponds to vMix Output 2, bus 2 to Output 3, and so on. vMix Output 1 is the main program output and cannot be reassigned by external applications like Mosart. **Info:** Newsroom tag available.
XPOINT	Crosspoint on the video switcher. **Info:** Newsroom tag available.

Item	Description
AUDIOLINK	Assigns a link group to this delegation. Please refer to Audio and Video Setup for the use of this feature. 
SET DELAY	Set the delay in frames of the key delegation when taking the template to Program. If the field is left blank, the key is set in Preview
TRANS DUR	If the vision mixer has the ability to mix on the AUX channels, this is where the mix rate (in frames) is set, *0* is a cut

Info: AUX transitions are only supported when using the GV CPL protocol.
Due to a limitation in the Grass Valley device, only *one* AUX bus can be transitioned at a time. This restriction comes from the switcher hardware, not from Viz Mosart.

Router source memory

Viz Mosart uses an internal AUX memory logic, where Mosart always stores the last crosspoint set on a specific AUX. If this AUXx is used as a *source* (rather than a destination) Mosart uses the source stored in memory. In the Template Editor, checking the Source drop-down list for a switcher crosspoint reveals at the bottom a list (that is automatically added appended) of AUXx sources. These refer to the internally stored AUX sources.

A similar logic applies to router crosspoints. However, the AUXx sources displayed in the switcher crosspoint Source drop-down, are *not* automatically shown in the router sources drop-down. Add them manually to the Router Source table in AVconfig, for the AUX memory logic to operate.

Info:

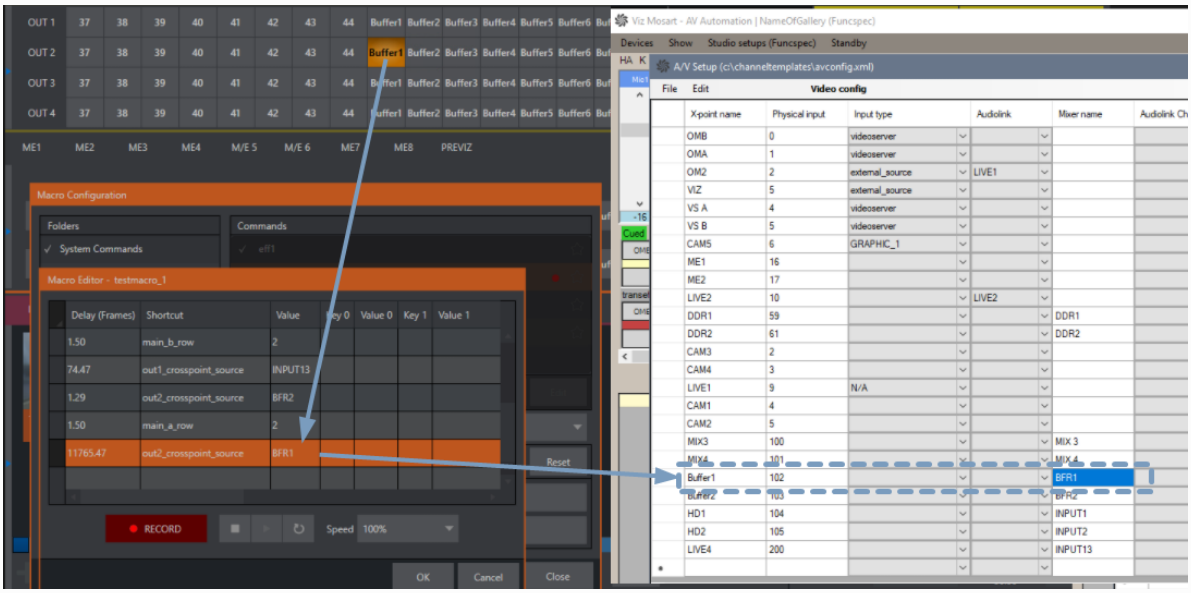
- The source named shall *not* contain special characters (for example dash **-** or underscore **_**).
- Naming in the form *AUX 9* or AUX9 (without space) is recommended.

- Using names like *AUX-9* or *AUX_9* will cause Mosart to ignore the memory logic and instead use the newly provided source identity.

Configuring an NDI Router (Viz Vectar Plus or Tricaster VMC)

Info: For NDI output crosspoints, you **must refer to actual crosspoint names recognized by VMC1** (and *not* any user-friendly names that would be mapped *to* the physical inputs).
These default naming on the NDI router side is typically:
- *INPUTn*
- *DDRn*
- *BFRn*
- *MIX n*.

The same naming must be used on the Viz Mosart side.
In **AV Automation > Devices > A/V Setup > Video Config > Mixer name column**, you must provide the hardcoded name associated with the user-friendly name of the crosspoint.
This name is found on the switcher side.
In the example below, the value of the literal crosspoint name is the router output found in the switcher's **Macro Configuration > Macro Editor** menu and added to AV Automation:



There is a new setting **UseOutputCrosspointNames** in the IP-protocol config VizrtIpSystemsVideoSwitcherConfig.xml set by default to expect the *crosspoint name*, instead of the Viz Mosart standard of using physical input naming.

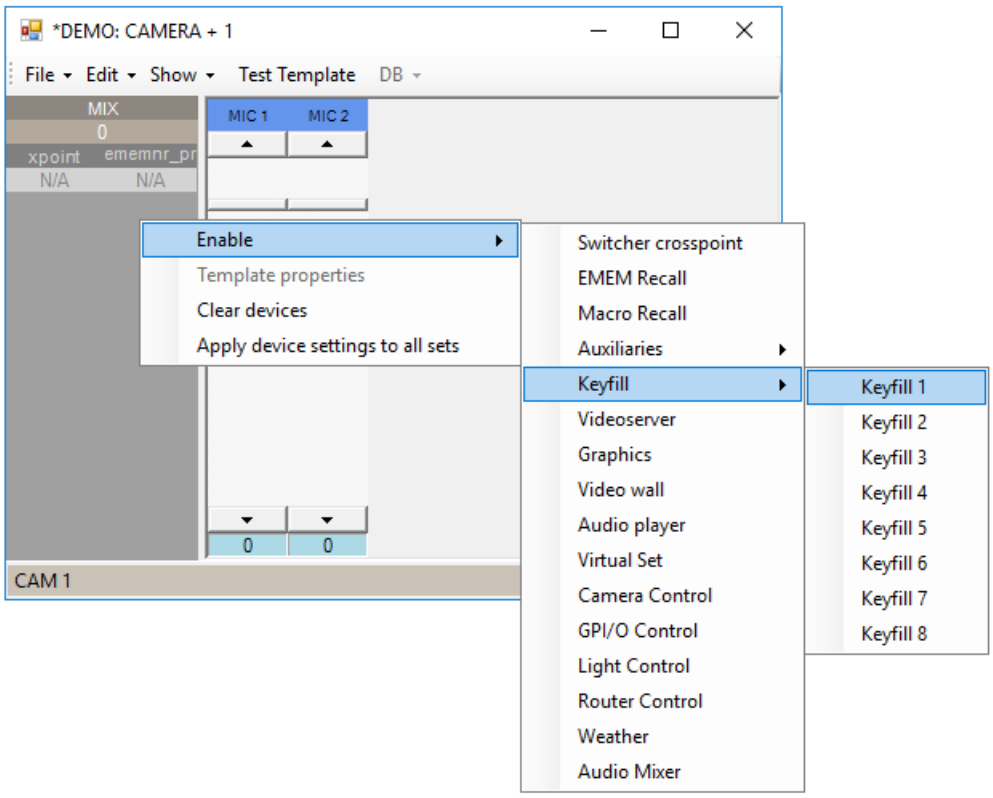
Keyfill: Video Switcher Key Bus Delegation

Video switcher key bus delegation enables routing of internal video switcher signals to the keyers on the different mixer effects of the switcher.

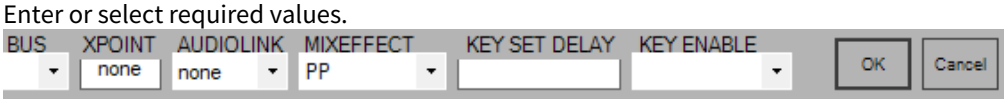
Info: For TriCaster video switcher supporting M/E effects with 4 layers, the device is named *Key/Layer*.

To enable Keyfill by bus delegation

1. Open **Template Editor**.
2. Navigate to **Enable > Keyfill > [Keyfill index]**



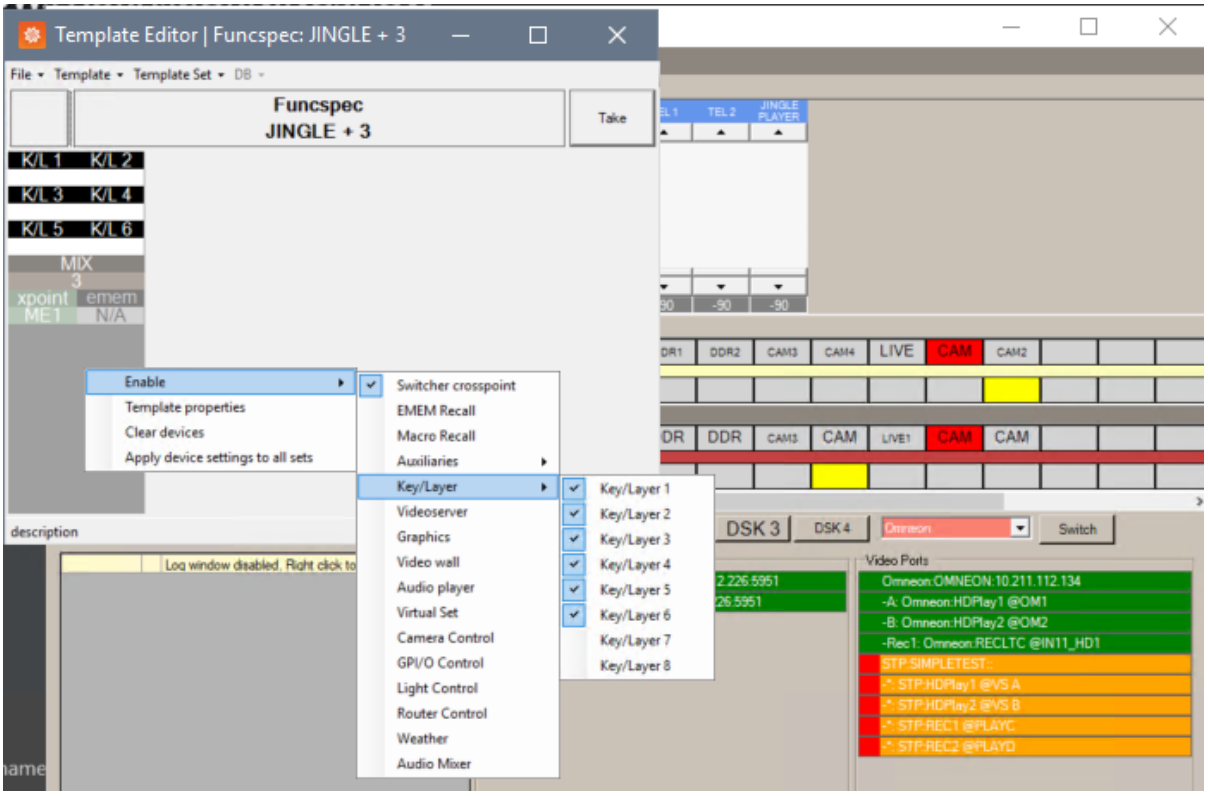
3. Enter or select required values.



To enable Keyfill by bus delegation for M/E effects with 4 layers

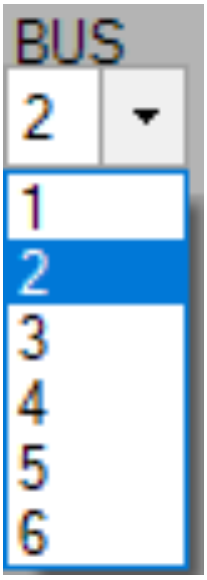
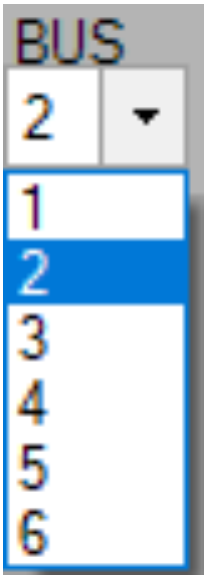
Devices like Vizrt's TriCaster family TriCaster Plus or VMC1 let you configure M/Es with effects supporting up to four primary sources/input layers (along with Key layers), called *A*, *B*, *C* and *D* buses.
Viz Mosart can control these buses from a template or keyboard shortcut when TriCaster is used as a vision mixer.

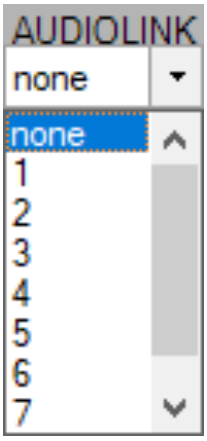
1. Open **Template Editor**.
2. Navigate to **Enable > Key/Layer > [KeyLayer index]**
   The Key/Layer device on a template can be used for VMC1 vision mixer to either set the crosspoint for a Key bus or to set the crosspoint for *A*, *B*, *C* and *D* buses of a M/E effect with 4 layers.

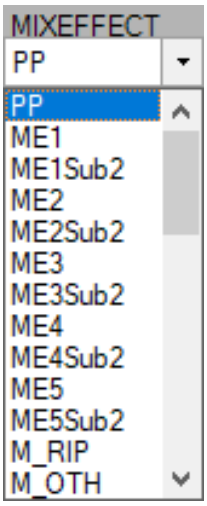
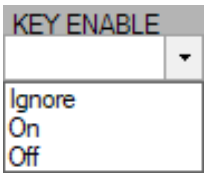


3. **Right-click > Device properties**.
   Select from the **BUS** drop-down menu:
   a. If the device is used for setting the crosspoint for a *key bus*, set the **BUS** drop-down menu to a number between *1* and *8* on the Key/Layer device properties.
   b. If the device is used for setting the crosspoint for a layer of a *M/E effect* with 4 layers, then choose one of the letters *A*, *B*, *C* and *D* from the **BUS** drop-down menu.

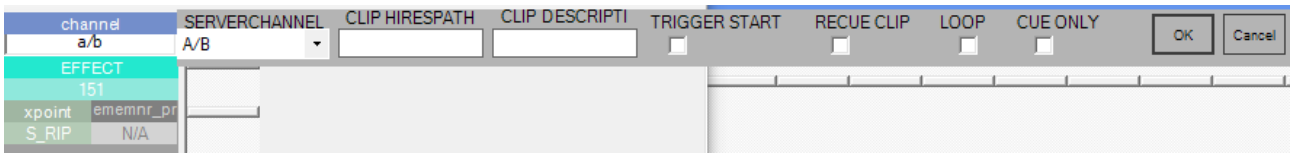
Guidelines for Keyfill Configuration

Item	Description
BUS	Selects the key bus on the selected ME.  If the video switcher supports M/Es with four layers (for example Vizrt's **TriCaster** family), four more buses are available corresponding to each layer, A, B, C and D: 

Item	Description
	If the switcher is **Ross Acuity**, the number of buses is *16* where the first 8 are used for the selected mixer region keyers and the next 8 buses target the second channel of the keyers. For example, *Bus 9* on *ME1* can be used to set the source on the second channel of *Key 1* of *ME1*. **Info:** Newsroom tag available.
XPOINT	Crosspoint on the video switcher. Default value: *none* **Info:** Newsroom tag available.
AUDIOLINK	Assigns a link group to this delegation. Please refer to Audio and Video Setup (previously *Linking video and audio sources)* for the use of this feature. 

Item	Description
MIXEFFECT	Select the ME where the key crosspoint should be set.  **Info:** Newsroom tag available. **Info:** For **ATEM**, both PP and ME1 map to the *same* physical ME1 on the switcher. PP targets downstream keyers (DSKs), while ME1 targets upstream keyers.
KEY SET DELAY	Set the delay in frames of the key delegation when taking the template to a program. If the field is left blank the key is set in preview.
KEY ENABLE	Key enable drop-down menu.  **Info:** Newsroom tag available.

Video Server



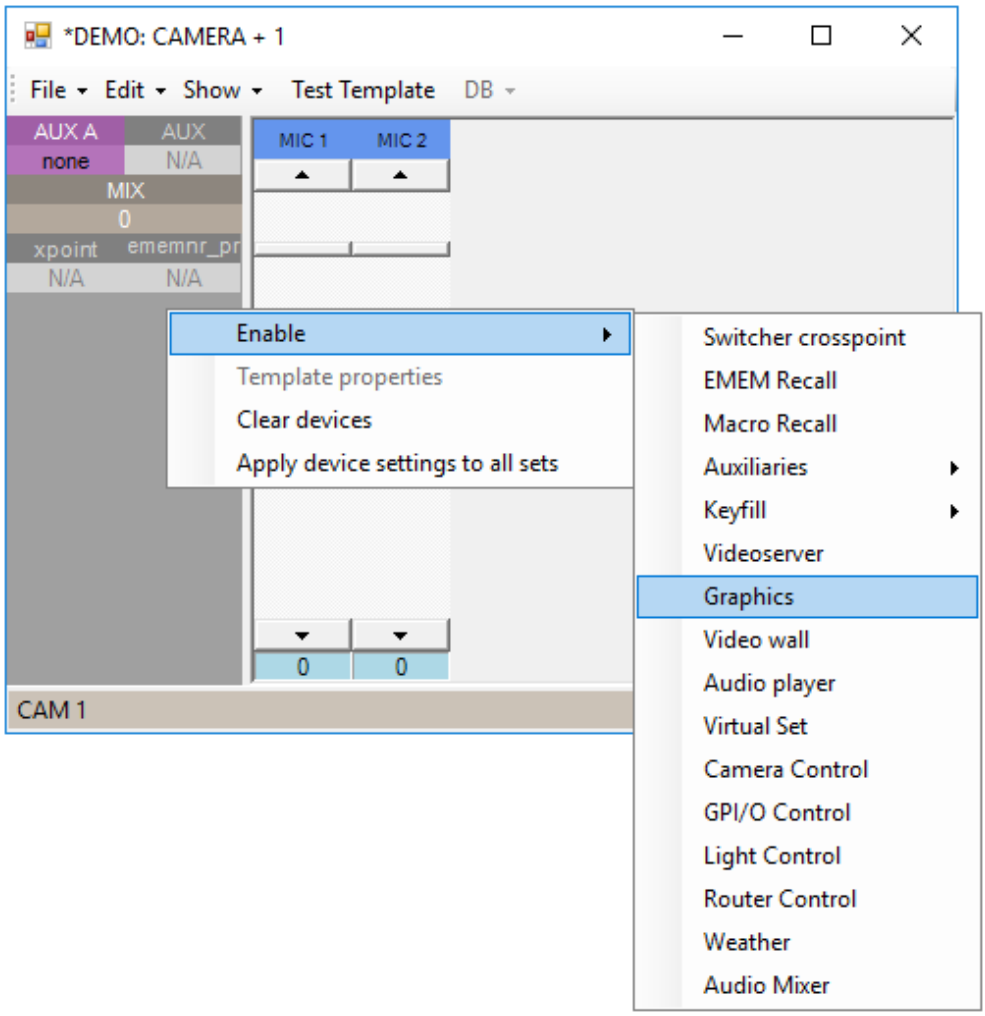
Guidelines for Video Server Configuration

Setting	Description
SERVERCHANNEL	The Mosart video port / port group where the clip is to be played out /recorded. For recording the corresponding video port has to be part of the “Rec” port group. Port groups are created as part of the video server configuration in menu **AV Automation > Devices > Preferences**. Configurations are described in Video Servers, section *Working with Mosart Port Configurations*. Notably, the defined virtual ports are among the available options. In addition, the non-empty groups (groups with at least one port) are available. If a group has the same name as one of its ports (as may be the case for the *Rec* and *P* groups, if defined), the *group* as such is not available, but the *port* is.
CLIP HIRESPATH	The (default) clip ID or name. **Info:** Newsroom tag available, meaning that a value from the NRCS will override the value specified here.
CLIP DESCRIPT	You can enter a Clip description, however a clip description from the NRCS or video server overrides this value. **Info:** Newsroom tag available.
TRIGGER START	Enables the Trigger start function from control command TRIGGER_START, a shortcut or continue point could be set to PLAY the Clip.
RECUE CLIP	In an A/B roll situation, when several Adlibs are played back-to-back, this setting determines how a clip will be restarted after it has been paused. For example, if Clip A is paused when Clip B is taken On Air, then next time Clip A is taken on the same channel, it can either be played from where it was paused, or be re-cued (played from the beginning). This is *only* used for AdlibPix templates.

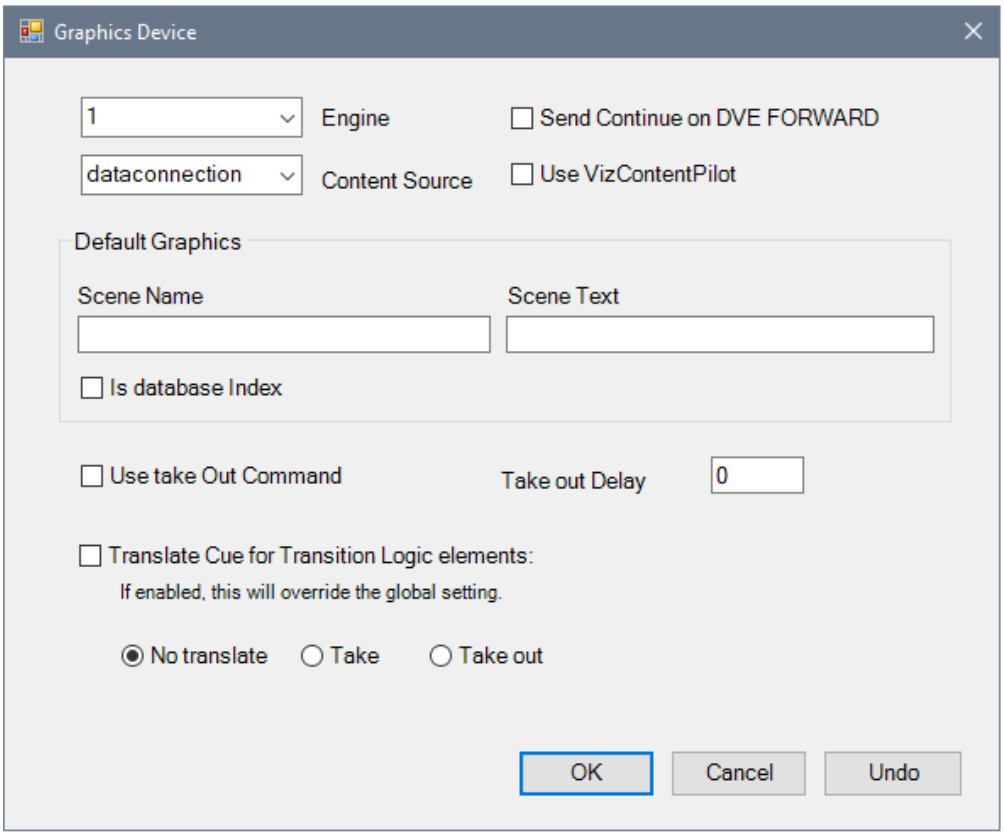
Setting	Description
LOOP	Sets the server port to loop the clip. Beware! Not all video servers can loop.
CUE ONLY	(Intended for *Accessory* and *Package* templates) Cues only the clip when the template is in Preview. The template will *not* start playing the clip when the template is taken.

Graphics

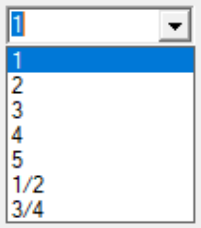
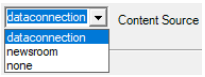
The Graphics device enables recall of graphic elements.



- Graphic elements are cued (loaded) in Preview and played (started) when taken to Program.

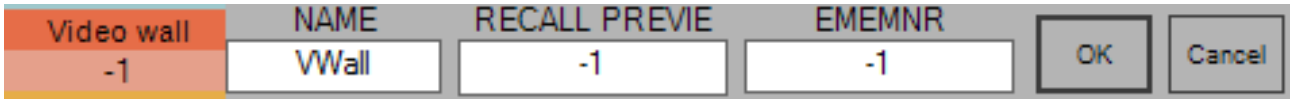


Guidelines for Graphics Configuration

Engine  	Selects the engine to operate on when using this template, based on your configuration. See section Graphics.
Content Source  	• **dataconnection:** Use content from the attached graphics database (only available when *Vizrt graphics* are used) • **newsroom:** Use content from the attached NRCS. • **none:** Use content written into the template.
Send Continue on DVE FORWARD	If selected, a continue command is sent together with a DVE Forward command.

Use VizContentPilot	Enable to run this element through a Viz Pilot client. This allows for OnLiveUpdate events in Viz Pilot data elements.
Scene Name	The default scene name is used, if not overridden by the NRCS. **Info:** Newsroom tag available.
Scene Text	The default scene text will be used, if not overridden by the NRCS. **Info:** Newsroom tag available.
Is database Index	
Use take Out Command	If selected, a take-out command is sent when the template status changes from On Air to Off Air.
Take Out Delay	Delay of sending takeout command after template status changes from On Air to Off Air.
Translate Cue for Transition Logic elements	If active, this setting overrides the global setting with same name set in AV Automation > Devices > Properties > Graphics > General > [*VIZRT*] > Miscellaneous. This setting only applies to transition logic for Vizrt full screen graphics. The Media Sequencer ignores standard cue command for these items. • **No translate**: Send cue command to the Media Sequencer (on Media Sequencer versions below 1.20 no cue will be performed). • **Take**: Use a Take command when cuing. • **Take out**: Use a Take out command when cuing.

Video Wall Register Recall



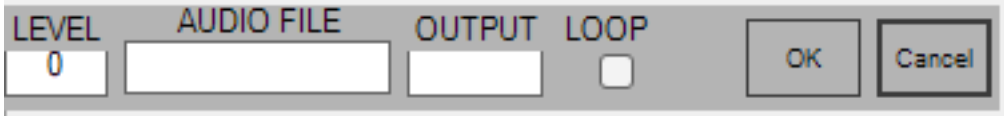
Setting	Description
NAME	Internal name of the shot.
RECALL PREVIE	For WATCHOUT, see notes below. For the other brands, this is the number of the recall to be taken when the template is cued in Preview.

Setting	Description
EMEMNR	For WATCHOUT, see notes below. For the other brands, this is the number of the preset to take when the template is taken in Program.

Info: Notes for **WATCHOUT**

- The fields RECALL PREVIE and EMEMNR should contain the command or sequence of commands to be sent when the template comes in Preview or Program, respectively.
- The general format is: `[load "show"] [` reset `run "timeline"]`
- Both `load` and `run` are optional, however at least one of them must be present. The show and timeline parameters may not contain double quotes.
- The `reset` value is optional. For more information see the connection string property *ResetAnyway*, described in the section Video Wall Connection Strings.
- In most cases, RECALL PREVIE contains `load "show"`, and EMEMNR contains `run "timeline"`.
- Commands supported: load, run, reset, halt, kill, gotoControlCue.

Audio Player



This description applies to the Viz Mosart Audio Player. For the TriCaster Audio Player and third party audio players, see under *Audio Player* in section Audio.

- **LEVEL:** This value *does not* affect the level! Instead, a value > -90 (the value must be an integer) *starts* playout, whereas a value <= -90 *stops* the playout.
- **AUDIO FILE:** Enter the audio file name here to play a fixed audio clip.
  Alternatively, enter a newsroom tag to get audio file name from the NRCS.

Info: All playable audio files must reside in the directory specified in the Mosart Audio Player.
For setup and configuration, see section *Audio Player* in AV Automation Devices (Audio).

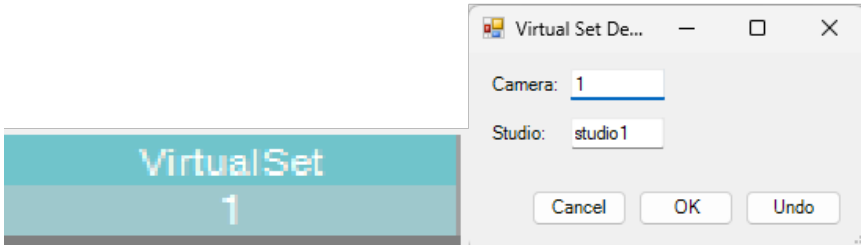
- **OUTPUT:** Choose the output on the Audio Player to be used. If no value is given, *Output 1* is used.
  This value will be overwritten if any (newsroom tag) value comes from the NRCS.

Info: *Outputs* are displayed on the Mosart Audio Player UI. See section *Audio Player* in AV Automation Devices (Audio).

- **LOOP**: When selected, at end the audio file always replays.

Info: Newsroom tags available for the three fields, LEVEL, AUDIO FILE and OUTPUT.

Virtual Set



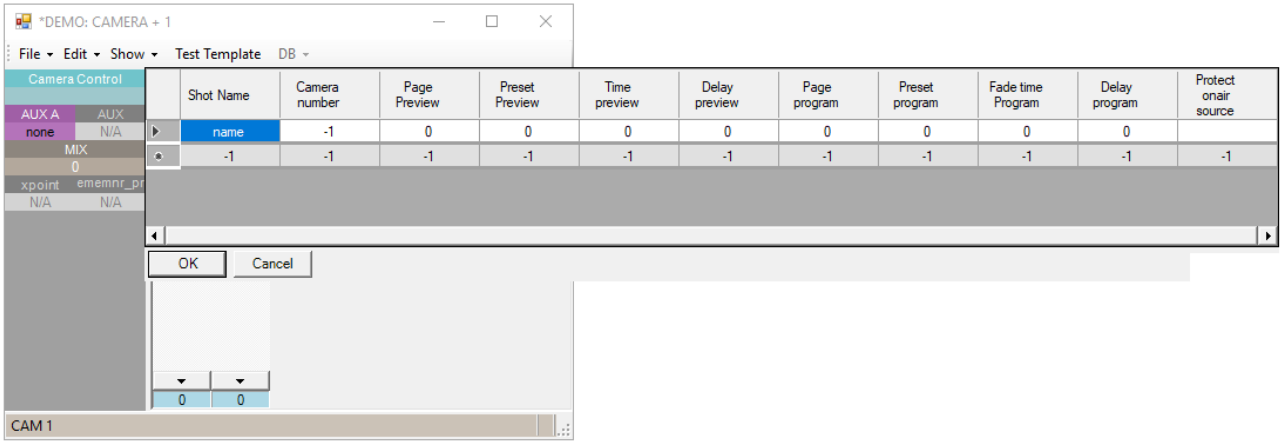
Viz Virtual Studio :

- **Camera :** Preset router index number to recall in Viz Tracking Hub currently loaded studio configuration
- **Studio :** Studio configuration name to recall in Viz Tracking Hub

Info: For Vizrt's virtual set manager, Viz Arc, please refer to section Viz Arc support within a Mosart show.

Robotic Camera Control

The camera control device allows for recalling camera shots (and pre-programmed moves) when the template is cued in preview or aired in program. The device editor supports salvo recalls of shots by adding multiple lines for each shot recall.



The Standard Robotic Camera Control Properties are described below, as well as exceptions for specific camera types.

- Standard Robotic Camera Control Properties
- Cambotics Properties
- Camerobot Properties

- Cinneo System Properties
- Fx-Motion Properties
- NDI PTZ Properties
- Panasonic Properties
- Technodolly Properties
- Shotoku TR_T Properties
- Telemetrics Properties
- Vinten 200 Properties
- Electric Friends Robotic Camera
- Setting up a Template with Camera Control Device Enabled

Guidelines for Standard Robotic Camera Control Configuration

Shot Name	Not used.
Camera number	• -1 : Shot is sent to all cameras that are not in standby. • >=0 : Shot is sent to the camera with the same number. The number refers to those given in the Cameras fields in AV Automation > Devices > Camera Robotics.
Preview	*The exact meaning and format of the* ***Preview*** *fields depend on the type of camera robotics. (See the sections below.)* These following three fields govern what happens when the template is taken in preview.
Page Preview	This field (together with Preset Preview below) determines the shot/move.
Preset Preview	This field (together with Page Preview above) determines the shot/move.
Time preview	This field (together with Preset Preview and Page Preview above) determines the duration of a move.
Delay preview	Not used.
Program	*The exact meaning and format of the* ***Program*** *fields depend on the type of camera robotics. (See the sections below).* These following three fields govern what happens when the template is taken to air.

Page program	This field (together with **Preset program** below) together determine the shot/move.
Preset program	This field (together with **Page program** above) together determine the shot/move.
Fade time Program	This field determines the duration of a move.
Delay program	Delay of move in Program (frames). **Info**: Only available for Vinten robotics systems
Protect onair source	Protected video switcher cross-point. If the video switcher cross-point is On Air, the shot/move recall will be ignored. For *Technodolly*, this protection can be confined to Preview and/or Program, and to Cuts and/or Moves.

Info: Newsroom tags are available on every column of the Robotic Camera Control device — Shot Name, Camera number, the Page, Preset, Time and Delay columns for both Preview and Program, Fade time Program, and Protect onair source. Right-click a cell and select **Add newsroom tag**; see Linking Device Properties and Newsroom Tags. These are free-text fields in the NRCS.

Cambotics Properties

- **Camera Number:** Set camera number to `1`.
- **Page Preview/ Page program:** Robotic camera number.
- **Preset Preview/ Preset program:** Shot number.
- **Time preview/ Fade time Program:** [Optional] Duration for move in deciseconds (10 deciseconds = 1 second).

Note: All parameters not defined specifically, *must* be set to *-1*.

Camerobot Properties

- **Page Preview/ Page program:** Name of shot's matrix in preview/program.
- **Preset Preview/ Preset program:** Shot's cell number in the selected matrix - the shot is adjustable from NRCS and GUI.
- **Time preview/ Fade time Program:** Duration of the move, in frames. A value of 0 sends a cut (*MATRIX_CUT_MPOS*) to the position; any other value sends a move (*MATRIX_MOVE_MPOS*). Adjustable from NRCS and GUI.
  Info: Newsroom tag available.
  Info: By default the value only selects cut or move, and Camerobot uses its own programmed move speed.

When **SendDuration** is enabled in *RoboticCameraConfiguration.xml* (see Robotic Camera Configuration Files), the value is also converted to seconds using the configured video frame rate and sent as a *<duration>* element, so the move lasts exactly that long. For example, 50 frames at 25 fps sends *<duration>2.00</duration>*. Combined with a newsroom tag on this column, the duration can be set per story from the NRCS.

Cinneo System Properties

- **Page Preview/ Page program:** Name of the camera to recall.
- **Preset Preview/ Preset program:** Name of shot to recall - adjustable from NRCS/GUI.
- **Time preview/ Fade time Program:** Desired moving time (seconds).

Fx-Motion Properties

- **Page Preview/ Page program:** Identifies the camera to recall (identical to *Camera Number* > 0), optionally followed by :category for temporary overriding any NRCS or GUI settings.
- **Preset Preview/ Preset program:** Name of shot or move to recall.
- **Time preview/ Fade time Program:** =0 cut, <0 ~ default time to shot, >0 ~ wanted time (frames).

NDI PTZ Properties

- **Camera Number:** A number as configured in the **Cameras** field for a robotic camera controller (see section Camera Robotics). This is used by Mosart to filter the shots for execution by a camera robotic controller.
  If value -1 is set, the shots will be executed for all controllers are not in stand-by, as configured in AV Automation Device Properties.
- **Page Preview/Page program:** A number as configured in the **Cameras** field for a robotic camera controller (see section Camera Robotics). This identifies the exact camera for which the shot will be executed.
  As long as the **Camera number** is greater or equal than 0, a number within the range of cameras supported for that controller should be set for either **Page Preview** or **Page Program**.
- **Preset Preview/Preset program:** Name of shot or move to recall.
- **Delay preview**: Not supported.
- **Delay program**: Delay of move in Program (in frames).

Panasonic Properties

- **Camera Number:** Identifies the Controller that the command is linked to.
- **Page Preview/ Page program:** Identifies the Camera Number to be controlled in Preview/Program.
- **Preset Preview/ Preset program:** Identifies the Preset Stored Move in Preview/Program.
- **Time preview/ Fade time Program:** Sets the Timed Delay in Preview/Program. Has no effect for Panasonic TCP.

Technodolly Properties

- **Page Preview/ Page program:** Identifies the camera, identical to *Camera Number* when >0.
- **Preset Preview/ Preset program:** Name of the move to recall.
- **Time preview/ Fade time Program:** =0 ~ go to start of move, ?0 ~ move.

Shotoku TR_T Properties

- **Page Preview/ Page program:** Page of shot to recall.

Telemetrics Properties

- **Page Preview/ Page program:** Preset page number to recall.
- **Preset Preview/ Preset program:** Preset number to recall.
- **Time preview/ Fade time Program:** =0 cut, <0 ~ default time to shot, >0 ~ wanted time (seconds).

Vinten 200 Properties

- **Page Preview/ Page program:** Name of the show to recall.
- **Preset Preview/ Preset program:** Name of shot or move to recall.
- **Time preview:** <0 ~ go to end position of a move, = 0 ~ cut to shot, >0 ~ go to start position of a move, or move to a shot (frames).
- **Fade time Program:** For moves, move forward or backward according to *Time preview*. For shots, = 0 ~ cut to shot, ?0 ~ move to shot (frames).

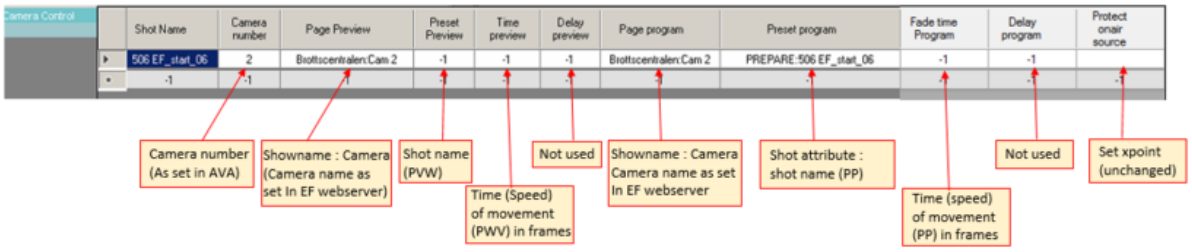
Electric Friends Robotic Camera

For Preview use:

- **Page Preview:** Show and Robotic camera name separated with a colon (e.g.: *ShowName:Cam 1*).
- **Preset Preview** [Optional]: Shot name. Shot attributes: prepare, prepareplay and loop. Separated with a colon (for example, *prepare:ShotName* and prepareplay:ShotName).
- **Time Preview** [Optional]: Duration for move in frames. The time is rounded up to the nearest second.

For Program use:

- **Page Program:** Show and Robotic camera name separated with a colon (e.g.: *ShowName:Cam 1*).
- **Preset Program** [Optional]: Shot name. Shot attributes: prepare, prepareplay and loop. Separated with colon (for example, *prepare:ShotName* and prepareplay:ShotName).
- **Fade time Program** [Optional]: Duration for move in frames. The time is rounded up to the nearest second.



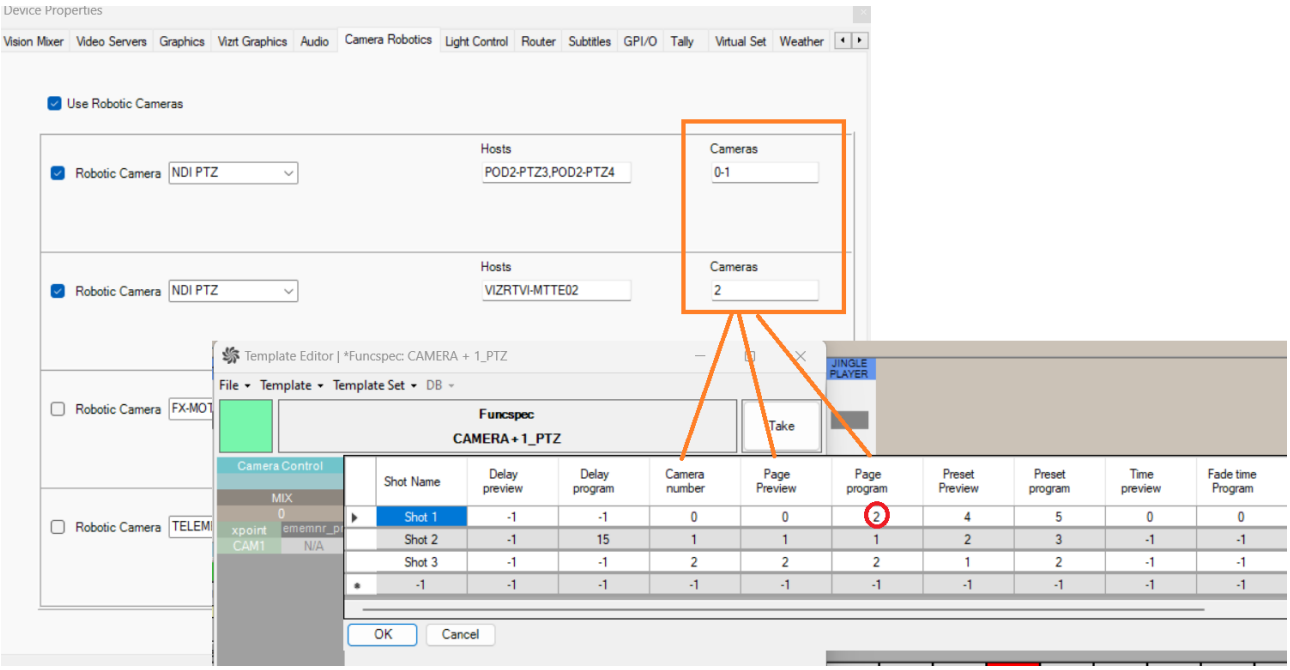
Example: Setting up a Template with Camera Control Device Enabled

This example demonstrates configuring a template for a Camera Control device using an NDI PTZ camera. Whilst the steps are specific to NDI PTZ cameras, a similar setup applies to other camera types.
Depending on the camera model and its supported features, slight adjustments in the configuration may be

required.
We assume the following setup is configured as described in section Camera Robotics (under *Device Properties*):

- POD2-PTZ3 and VIZRTV-MTTE02 are connected.
- POD2-PTZ4 is currently disconnected.

On a Mosart template, we have the following shots for a *Camera Control* device:

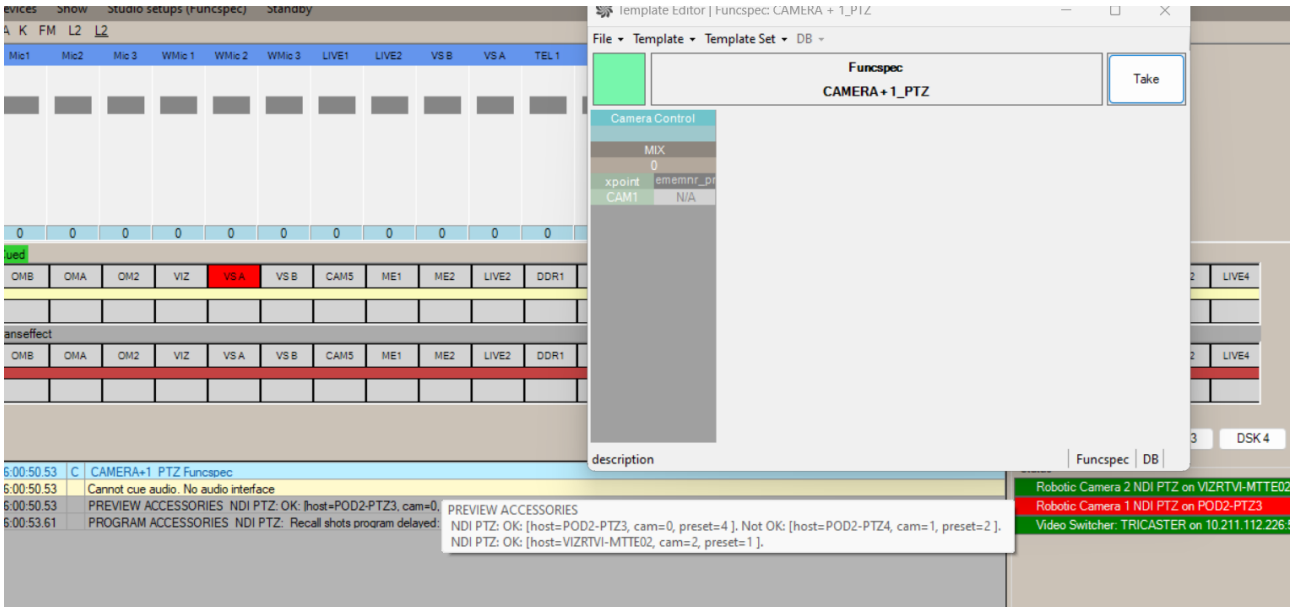


In the screenshot above, based on the numerical input **Cameras**, shots *1* and *2* will be executed by the first robotic camera controller, while *shot 3* will be executed by the second robotic camera controller.

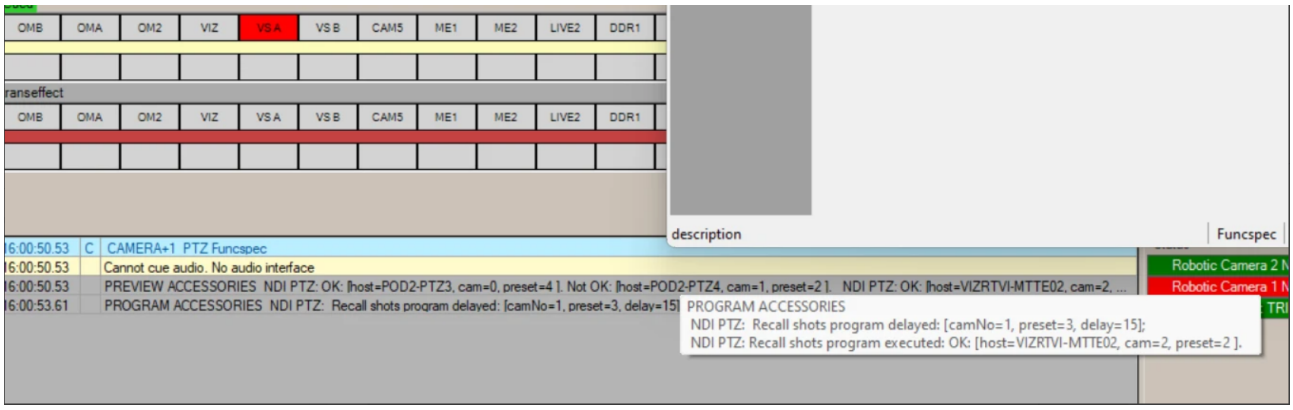
- **Shot 1:**
  - **Cameras number** is set to *0*, meaning the shot will be applied to the first controller configured, as 0 is within the range 0-1 set in the **Cameras** field.
  - **Page Preview** is set to *0*, meaning *Preset 4* (in field **Preset Preview**) will be applied to camera *POD2-PTZ3* in Preview.
  - **Page program** is set to *2*, which is outside the range 0-1, so the preset will *not* be applied in Program.
- **Shot 2:**
  - **Camera number** is set to *1*, meaning the shot will also be applied to the first controller, as 1 is within the range 0-1 set in the field **Cameras**.
  - **Page Preview** is set to *1*, meaning *Preset 2* will be applied to camera *POD2-PTZ4* in Preview.
  - **Page Program** is set to *1*, meaning *Preset 3* will be applied to camera *POD2-PTZ4* in Program, but with a delay of 15 frames.
- **Shot 3:**
  - **Camera number** is set to *2*, meaning the shot will be applied to the second controller configured, as the number matches the value set in the field **Cameras**.
  - **Page Preview** is set to *2*, meaning *Preset 1* will be applied to camera *VIZRTV-MTTE02* in Preview.
  - **Page Program** is set to *2*, meaning *Preset 2* will be applied to camera *VIZRTV-MTTE02* in Program.

Copyright © 2026 Vizrt

In other words, the following moves were executed in Preview (note that camera *POD2-PTZ4* is disconnected):

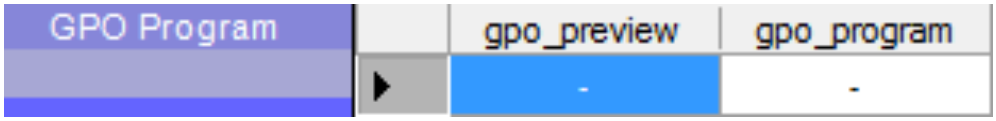


and the following moves were executed in Program (again, note that *POD2-PTZ4* is still disconnected, the shot will not be executed. This event also appears in the technical logging):



GPO

The GPO editor is used for sending GPOs to external equipment.

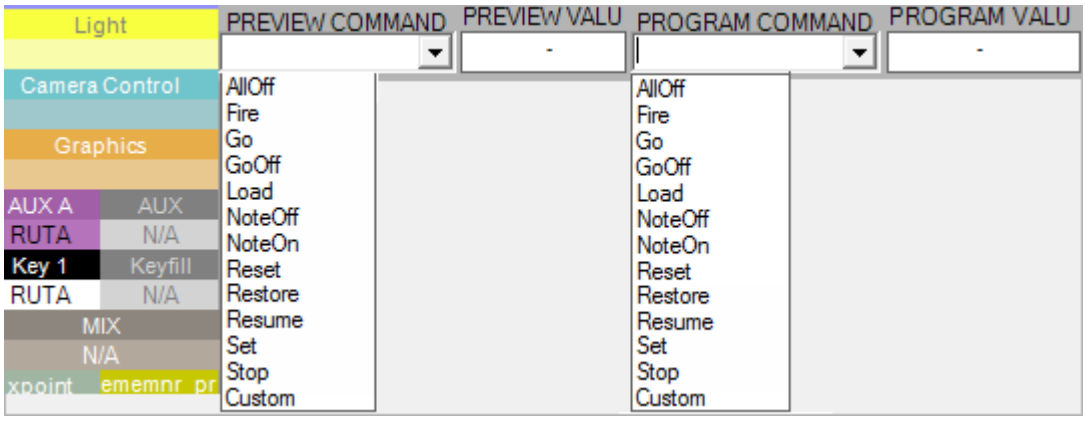


- The editor supports multiple GPO sends.
  Add a new line in the editor for multiple GPO sends.

> **Tip**: For GPO values, refer to section GPIIO.

- **gpo_preview:** GPO to send when a template is cued in Preview. **Info:** Newsroom tag available.
- **gpo_program:** GPO to send when a template is taken in Program. **Info:** Newsroom tag available.

Light Control



> **Info:** The commands to use are driver-dependent and may vary.

Guidelines for Light Control Configuration

Option	Description
PREVIEW COMMAND	Command to use for the light mixer cue to be recalled when template is cued in Preview. **Info:** Newsroom tag available.
PREVIEW VALUE	Light mixer cue to use when template is cued in Preview. **Info:** Newsroom tag available.

Option	Description
PROGRAM COMMAND	Command to use for the light mixer cue to be recalled when a template is taken in Program. **Info:** Newsroom tag available.
PROGRAM VALUE	Light mixer cue to use when a template is taken in Program. **Info:** Newsroom tag available.

Preview Command and **Program Command** options are available from the drop-down list.

For OSC drivers (grandMA3 or ETC Eos Family) messages will be parsed from *Custom* command text. See OSC parsing below.

Command	Parameter count
AllOff	<none>. For details, see *Set* below.
Fire	1.For details, see *Set* below.
Go	cue number. For details, see *Set* below.
GoOff	cue number. For details, see *Set* below.
Load	cue number. For details, see *Set* below.
NoteOff	numeric note # (Std. MIDI command). For details, see *Set* below.
NoteOn	numeric note # (Std. MIDI command). For details, see *Set* below.
Reset	<none>. For details, see *Set* below.
Restore	<none>. For details, see *Set* below.
Resume	variable. For details, see *Set* below.

Copyright © 2026 Vizrt

Command	Parameter count
Set	The value of SET is presented below in the context of Midi Show Control Specifications (RP-002-014_v1-1-1_MIDI_Show_Control_Specification_96-1-4.pdf available at https://midi.org/specs)

Hex value	Command	Number of data bytes	Recommended minimum sets
00	reserved for extensions		
01	GO	variable	1 2 3
02	STOP	variable	1 2 3
03	RESUME	variable	1 2 3
04	TIMED_GO	variable	2 3
05	LOAD	variable	2 3
06	*SET*	*4 or 9*	*2 3*
07	FIRE	1	2 3
08	ALL_OFF	0	2 3
09	RESTORE	0	2 3
0A	RESET	0	2 3
0B	GO_OFF	variable	2 3

Info: You can include an optional time value (key/value[/time]), in the form: key=value@hh:mm:ss:ff
If unexpected program conditions occur, use a (dummy) time parameter.
For example, *511=22@00:00:00:00*, meaning set key 511 to the value 22 at time (hh:mm:ss) 23:59:59, frame 24.

Stop	cue number. For details, see *Set* above.

Command	Parameter count
Custom	SysEx bytes Send multiple custom SysEx bytes in the format `\xnn` where `nn` is a numeric value (`0-255`). The global prefix and postfix set in the device config is used in the message (for example, `abc\x20def` will send seven bytes as a SysEx message). For details, see *Set* above.
TimedGo	- (unsupported). For details, see *Set* above.

OSC message parsing

For OSC-controlled lighting consoles, the Custom command type must be used for both Preview and Program; the string then added in the “Value” field should then be written according to the specification of the controlled device (grandMA3 or ETC).

grandMA3

1. The text before the first comma will be taken as the OSC address; for example: [*/Page1/Fader201*]
2. If there’s more text, then then next part until a comma will be interpreted as OSC type tags; for example: [*iTs*]
3. The rest of the text value will be interpreted as a list of OSC arguments, separated by commas; for example: [*5, text*]
4. Quoted texts will be parsed as one argument. Quotes can be escaped with “\“, and backslashes can also be escaped by “\“. Example: [*5, text, “2 words“, “text with \\ backslash“*]
5. For an OSC type that does not require an argument (for example *True, False*), there should not be any.
6. The number of provided arguments must be equal to the number of provided type tags (that require an argument).
7. The arguments must be parsable as their OSC type.

Supported OSC Types

- int32 (i)
- float32 (f)
- OSC-string (s)
- True (T)
- False (F)
- Null (N)
- Impulse (I)

Examples (for illustration only; always refer to MA documentation):

/cmd, s, Echo word
/cmd, s, "Echo \",1\""
/Page1/Fader201, i, 100
/someAddress, sis, Flash, 1 , Strobe 1 Cue 1

ETC Eos family

1. The text before the first equals sign will be taken as OSC address; for example: [/eos/group/1/param/pan/tilt]
2. The rest of the text value will be interpreted as a list of OSC arguments, separated by commas; for example: [5, 10]
3. Quoted texts will be parsed as one argument. Quotes can be escaped with “\“, and backslashes can also be escaped by “\“. Example: [*5, text, “2 words“, “text with \\ backslash“*]
4. Arguments will be parsed first as int32. If that fails, then float32, and if that also fails, as string.

Supported OSC Types

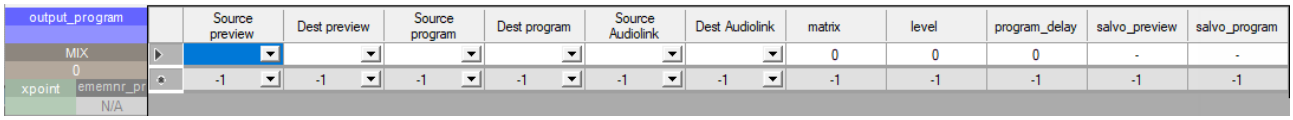
- int32 (i)
- float32 (f)
- OSC-string (s)

Examples (for illustration only; always refer to ETC documentation):

/eos/preset/fire=1
/eos/group/1/param/pan/tilt=90,75
/eos/cmd=”Chan 1 At 75"

Router Control

The router control device allows setting crosspoints for source/destination pairs when the template is both cued in Preview and aired in Program.



- Add new lines in the router device editor to support setting multiple crosspoints.

Guidelines for Router Control Configuration

Item	Description
Source preview	Source to route when a template is cued in Preview.
Dest preview	Destination of routing when a template is cued in Preview.
Source program	Source to route when a template is aired in Program.
Dest program	Destination to route when a template is aired in Program.
matrix	If the router system supports multiple matrices, select the matrix to use here.

Item	Description
level	• **0:** route on all levels. • **1:** video routing only. • **2:** audio routing only. • **3:** GPI/O routing only
program_delay	Delay, in frames after the template has been taken to the air, for 'program' routing to happen.
salvo_preview	Salvo name to recall when a template is cued in Preview.
salvo_program	Salvo name to recall when a template is aired in Program.

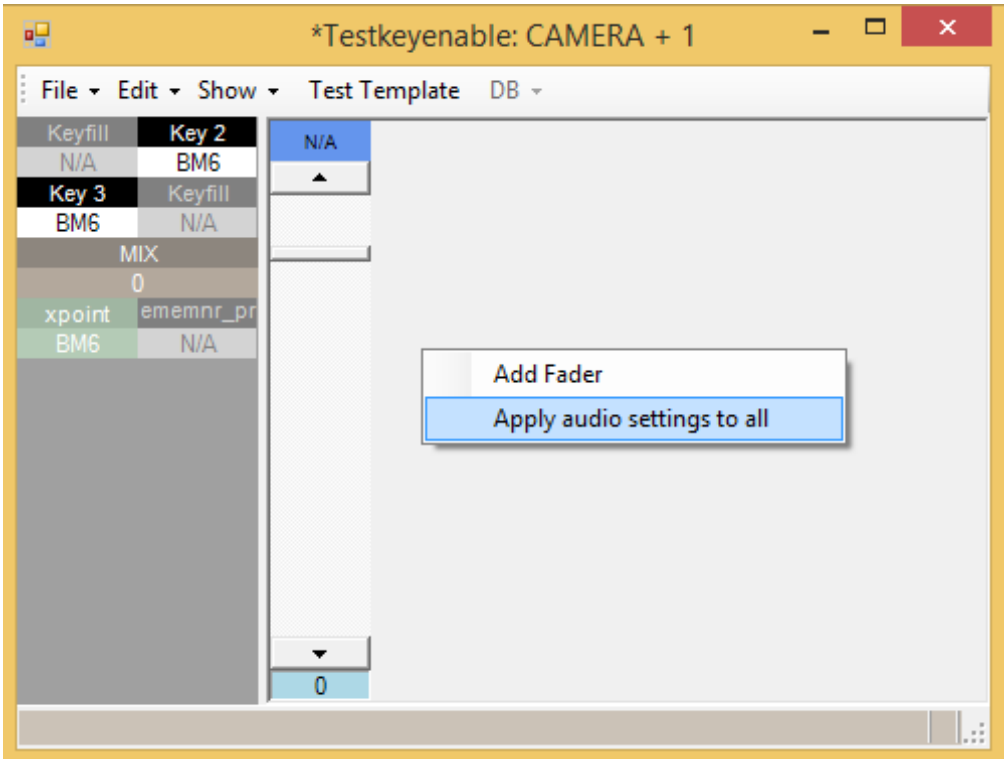
Info: Newsroom tags are available on every column of the Router Control device. *Source preview*, *Dest preview*, *Source program* and *Dest program* offer the configured router sources and destinations as a pick list; *matrix*, *level*, *program_delay*, *salvo_preview* and *salvo_program* are free-text fields.

Audio Faders

Adding, removing and configuring audio faders

To add audio faders

1. Add audio faders to the template by right-clicking in an empty part (large grey area to the left) of the audio fader area and selecting **Add fader**.
2. Set the audio mixer crosspoint from the drop-down menu by clicking the label at the top of the fader.



Note: Clicking **Apply audio settings to all** applies all the changes to all templates of the same type and variant throughout all template sets.

To set audio fader level

- The **main level** for the fader is set by dragging the notch or clicking the arrows.
  The default fader level when adding a new fader is `0` dB.
- To set **level 2** of the fader press **SHIFT** while dragging.
  A tool-tip opens to signal the editing of the level 2 fader level.
- To set **level 3** of the fader press **SHIFT+CTRL** while dragging.
  A tool-tip opens to signal the editing of level 3 fader level.
  Level 3 audio can be used with VOICEOVERS, when the AUDIO control command FADE_DOWN_SECONDARY_AUDIO is called.
  Video server faders are then faded to this level 3 (when set), instead of being muted.

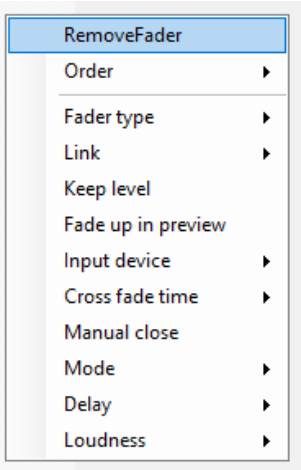
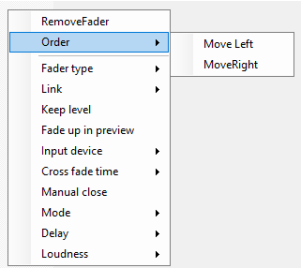
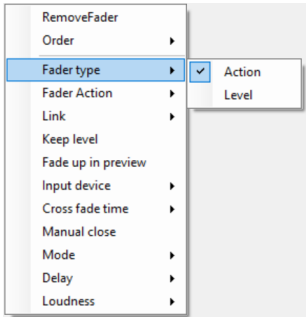
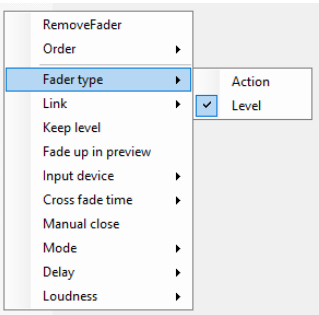
To remove a fader

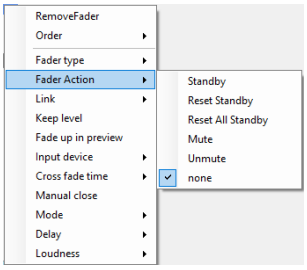
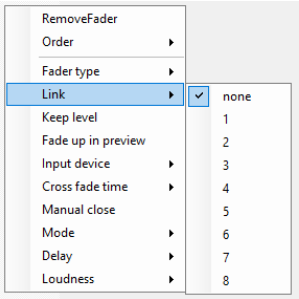
- Right-click over the fader and select **Remove fader**.

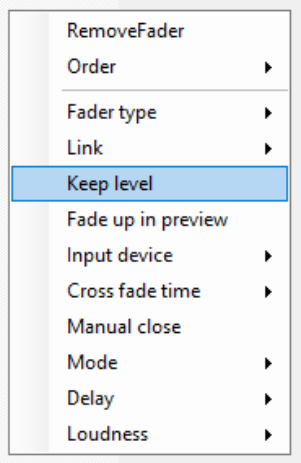
To configure a fader

- Right-click over the fader to open the context menu and follow the *Guidelines* below.

Guidelines for Audio Fader Configuration

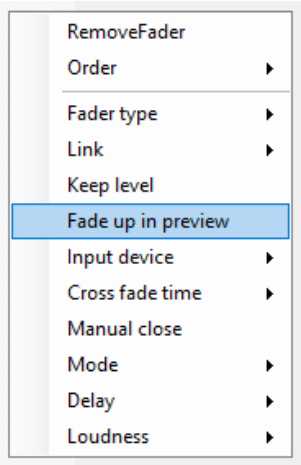
Remove Fader	Removes the fader from the template. 
Order	Rearranging the order: • *Move Left* • *Move Right* 
Fader type	**Action:** When a fader control is set as an *Action* fader, the controls define whether the fader is *enabled* or *disabled*. **Level:** For standard audio operation.  

Fader Action	 • **Standby**: Fader is placed in standby (A fader in standby ignores any commands sent to it via a template) • **Reset Standby**: Fader is placed in a non-standby status • **Reset All Standby**: All faders are placed in a non-standby status • **Mute**: Mute the fader • **Unmute**: Unmute the fader • **none**: Take no action
Link	Assigns a video/audio link group to this fader. • *none* • *1* • *2* • *3* • *4* • *5* • *6* • *7* • *8* 

Keep level	Keeps the fader level until manually changed. Enabled to let the automation open the audio fader, leaving it open until is is closed from another template, or the Viz Mosart client operator closes it with the **FADE_OUT_KEEPS** control command (recalled from an **Audio** keyboard shortcut). 

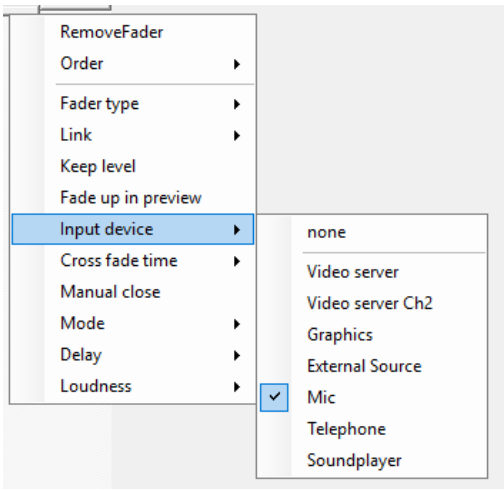
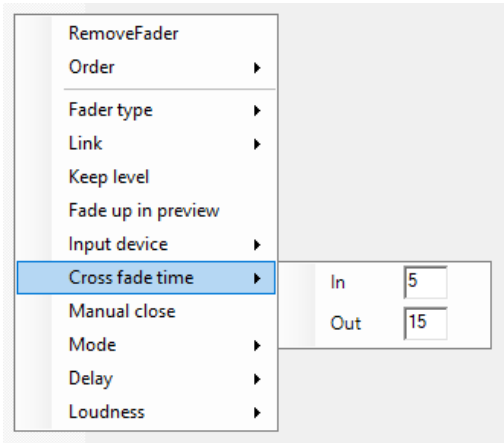
Fade up in preview

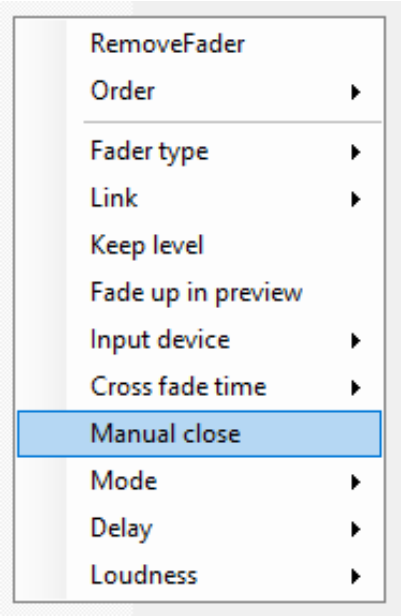
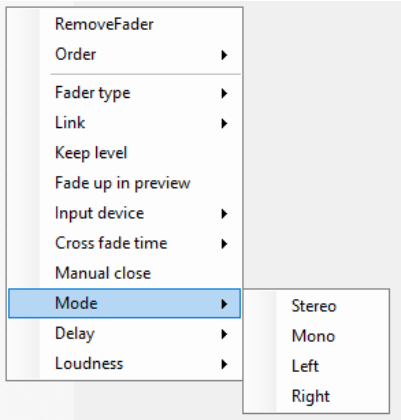
Fades up the audio in preview, when camera is in program.
To let the audio fader open when the template is cued in program, set to *enabled*.

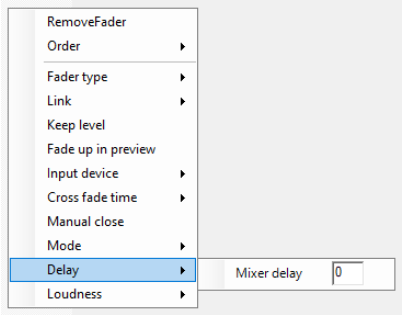
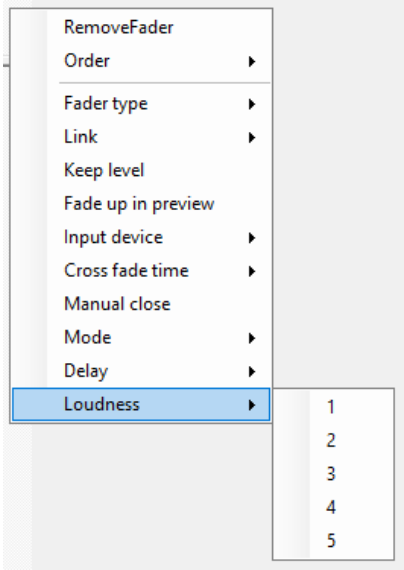


Enabling this functionality only works with specific template combinations, as shown below:

Program	Preview (with Fade up in preview enabled)	Result
CAMERA	LIVE	✔
	DVE	✔
	TELEPHONEINTERVIEW	✔
	PACKAGE	✔
	Other Templates	✘
Other Templates	PACKAGE	✔
	Other Templates	✘

Input device	Sets the type of the audio fader. The setting is used in combination with primary template type and On Air status to identify which fader should be treated as secondary audio. This special handling is applied to the types below: • **none:** No special treatment. ---------- • **Video server:** The fader controls a video server channel audio output. • **Video server Ch2:** The fader controls a second video server channel audio output. • **Graphics:** The fader controls audio output from the Graphics Engine. • **External source:** The fader controls an external audio source. • **Mic:** The fader controls a studio microphone. • **Telephone:** The fader controls a telephone hybrid. • **Soundplayer:** The fader controls the Mosart Audio Player output. 
Crossfade time	Sets the in and out crossfade time (in video frames) for the fader. • *In* • *Out* 

Manual close	Select to ignore sending Close-level commands to audio mixer. When selected, sending the Take out for the fader at the end of the element is ignored; The fader has to be manually taken down. 
Mode	Only for Studer Vista and Lawo! Sends the chosen value to the mixer board for the audio fader. • *Stereo (not for LAWO EMBER+)* • *Mono (not for LAWO EMBER+)* • *Left (LAWO EMBER+: Stereo faders only)* • *Right (LAWO EMBER+: Stereo faders only)* 

Delay	Only for Studer Vista, Studer 3000 using Ember and Lawo RMNOPL! Gives a delay value (in milliseconds) for that audio fader that will be sent to the audio board. • *Mixer delay*. Default: 0 
Loudness	Sends chosen recall value to send to the Jünger audio processor. • *1* • *2* • *3* • *4* • *5* 

Generic REST commands

See Controlling a REST-based Driver.

9.3.4 Newsroom Tags

Newsroom tags are the mechanism for transferring data values in a *configurable* form, from the NRCS to Viz Mosart. For example, a template for playing a clip can have a newsroom tag for the *clip name*. The journalist specifies the clip name in their NRCS story item. When the story item is taken, this clip name is then automatically transferred to Viz Mosart, where the correct clip is played-out.

A Newsroom tag has:

- A *name*
- Optionally, a *value*
- Optionally, a *list* of permissible values. If a list is given, the tag value, if any, will be in the list.
  This list will be one of those defined in AV Automation, section Audio and Video Setup:
  - Video config
  - Audio config
  - Effects
  - Router sources
  - Router destinations.

Note: Invalid characters in XML 1.0 are also invalid characters when used with ActiveX and newsroom tags. For more information, see the XML 1.0 documentation:
https://web.archive.org/web/20140228010526/http://validchar.com/d/xml10/xml10_namestart
https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references
https://en.wikipedia.org/wiki/Valid_characters_in_XML

The basic workflow for *one* tag is:

1. In the Template Editor, a newsroom tag is added to a template field. (See Adding a Newsroom tag to your Viz Mosart Template below).
   The tag *name* is specified.
   If the template field has an associated list of values (as listed above), that list will be the tag list, too. If a value is given for the template field, that value will become the tag value, too.
2. In the ActiveX in the NRCS, when inserting the template into a story item in a rundown, the tag value may be changed or set (depending on whether a value was given in the previous step).
3. When the rundown is loaded into Viz Mosart, the newsroom tag value, if any, is *automatically* assigned to the template field, overriding any value set in the Template Editor.

Note: Step [**1**] may be repeated, adding *several* tags to a single template. Then step [**2**] may be repeated in the ActiveX, and step [**3**] will automatically be repeated, too.
If in step [**1**], the *same* tag name is specified for several template fields (*in the same template*), only *one* newsroom tag will be added, so only one value need be supplied (for this tag; there may be others) in step [**2**].
In step [**3**], this single value will be assigned to all template fields (*in that same template*) having the newsroom tag with this name.

- (Optional) If the newsroom tag shows up in the GUI Program window or Preview window (Refer to the explanations for **Program** window and **Preview** window in the *Viz Mosart User Guide*, topic *Main User Interface*, section *Key Select*), the value may be modified there.

The remainder of this section deals with:

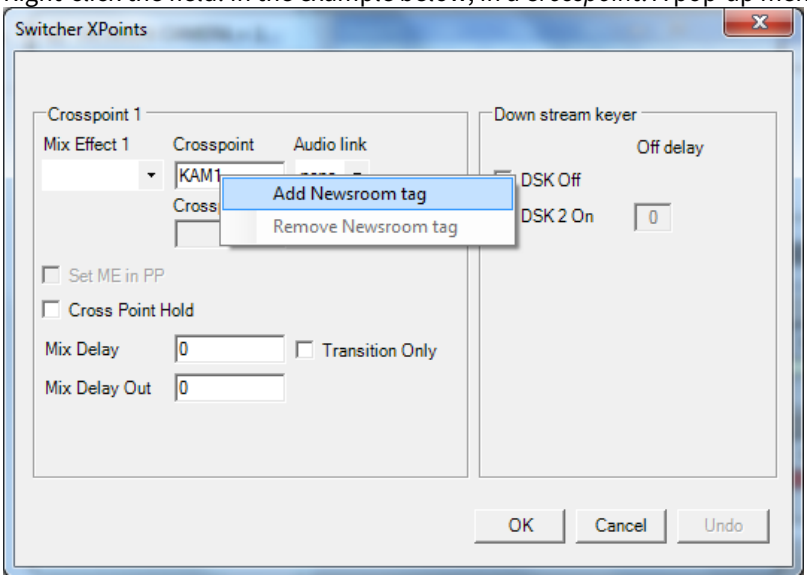
- Adding a Newsroom tag to your Viz Mosart Template
- Newsroom Tags Order
- Advanced Features for Specific NRCSs
- Inheriting Newsroom Tags (applies to MOS-based NRCS s only)

Working with Newsroom Tags

Adding a Newsroom tag to your Viz Mosart Template

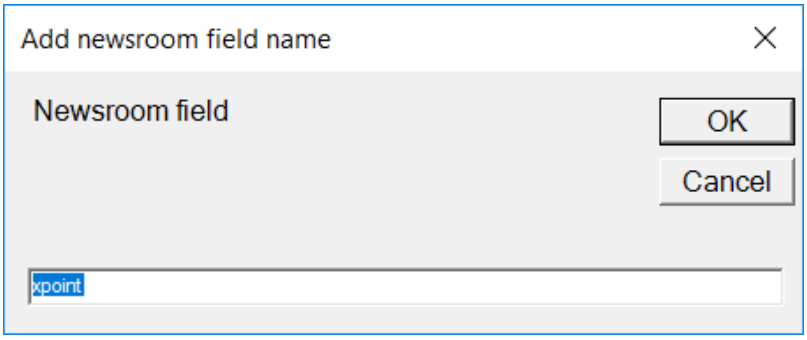
To Add a Newsroom tag

1. Refer to section Template Device Functions, which lists the properties of the device where you want to add the newsroom tag.
2. Right-click the field. In the example below, in a *Crosspoint*. A pop-up menu appears:



> **Note**: If the field already has a newsroom tag, there will instead be a menu option *Change Newsroom tag*, and the *Remove* option will be enabled.
> Not all template fields support newsroom tags. When right-clicking an unsupported field, a pop-up menu may or may not appear. If it does, it will not contain an *Add Newsroom tag* option.

3. Select **Add Newsroom tag**.
   A dialog menu appears:

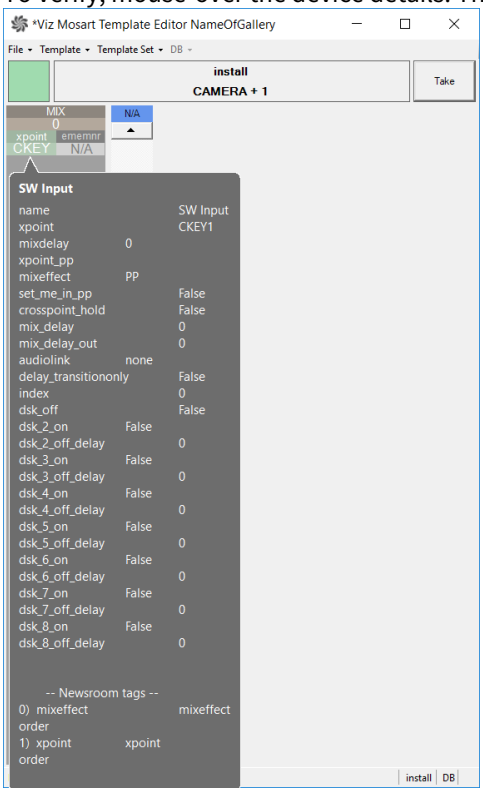


A default name may be given.

4. Enter or change the newsroom tag name, and click **OK**.

> **Note:** When naming a newsroom tag, avoid using any *special characters*.

6. Close the **Device Properties** menu (in the example above, *Switcher XPoints*).
7. To verify, mouse-over the device details. The Newsroom tag(s) are presented at the bottom of the panel.

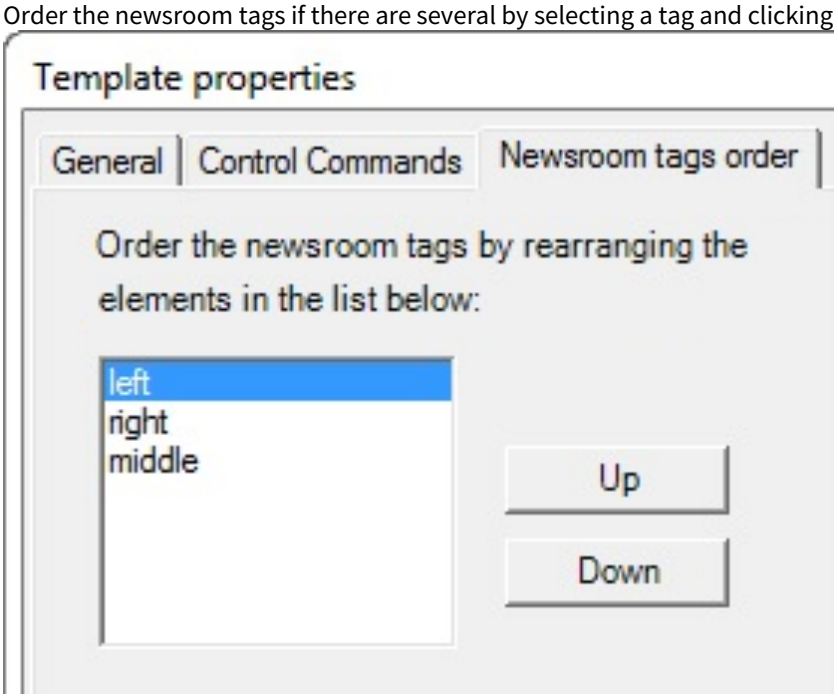


Newsroom Tags Order

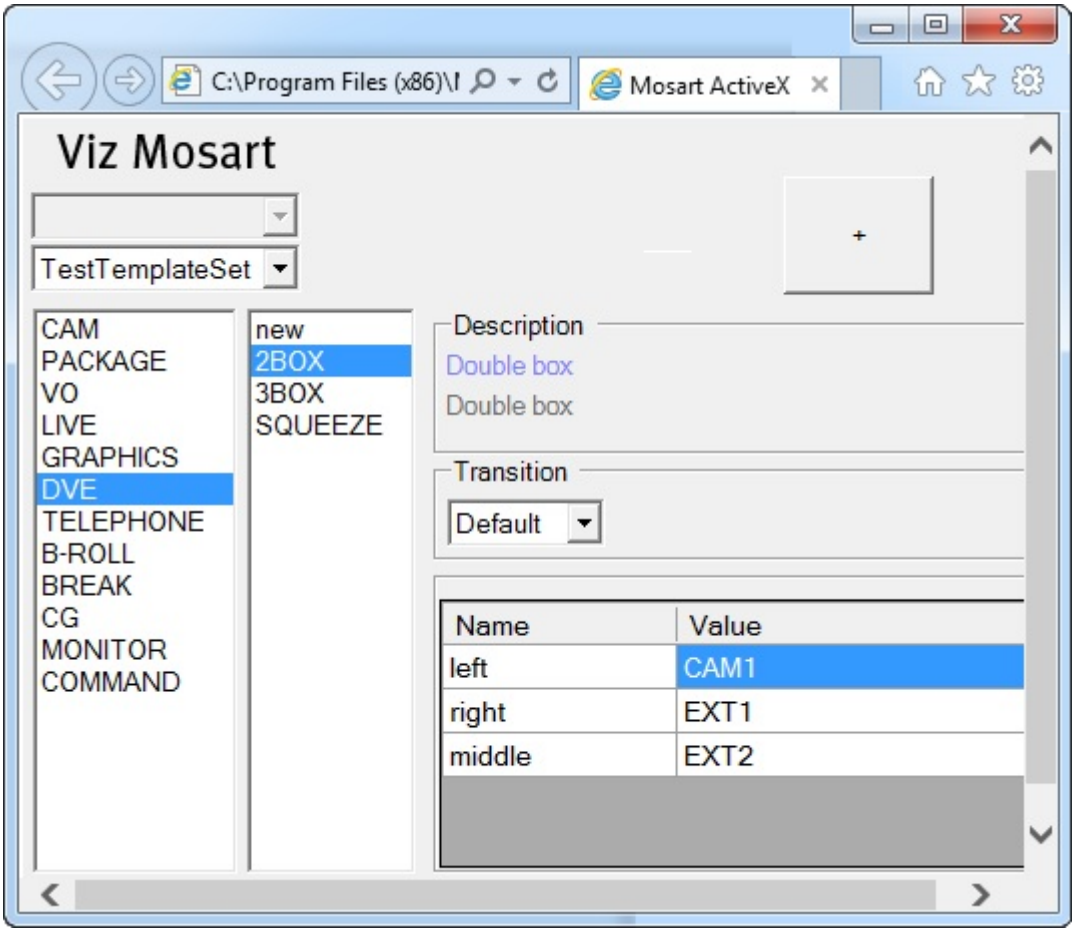
If a template has more than one newsroom tag, the order in which the tags are shown in Viz Mosart's ActiveX, may be changed in the Template Editor.

To set newsroom tag order

1. In the **Template Editor**, open the target template.
2. Open **Template properties** and select the **Newsroom tags order** tab.
3. Order the newsroom tags if there are several by selecting a tag and clicking **Up** or **Down**.



4. To verify, open the Activex in the NRCS, and observe that the tag order is as set in the previous step:



Advanced Features for Specific NRCSs

To Use a Column in ENPS as a Newsroom Tag

You can use a column value in ENPS as a newsroom tag in a Viz Mosart Template.

1. Add the column in ENPS, by using the TV station-specific routine to add a column in ENPS.
   All the current fields are visible in **System fields->ENPS fields**.

> **Usage Notes**:
> - The id of the field and the newsroom tag in Viz Mosart must be named *identically*.
> - Naming is case-sensitive.
> - Ensure that **Mos Send** is checked.
> - **Type** must be *Script+RO Column*.

2. Restart the ENPS client to add your new column in the layout.
3. Add the column as a story variable in Viz Mosart config:

File Edit **Story External Metadata**

	mostagname	Mosart Action	Format Type	mos_value	action_value
	inp1	item_variable	String		inp1
	inp2	item_variable	String		inp2
	inp3	item_variable	String		inp3
	inp4	item_variable	integer		inp4

4. In Viz Mosart **Story External Metadata**
   a. Add the new ENPS field as, for example, an *item_variable*.
   b. There is also a *story_valuable* available, but using this action value affects *all* items in the story (the newsroom tag binds to all primaries in that story).
   c. The target here should be the *first primary item* in a story.
5. Restart all Viz Mosart server applications to apply settings.
6. Add the newsroom tag to required template, a standard operation as described above in Adding a Newsroom tag to your Viz Mosart Template.

Inheriting Newsroom Tags (applies to MOS-based NRCS s only)

For MOS-based NRCS, Viz Mosart offers a method where a newsroom tag need only to be set once within a story.

> **Note:** If, for ENPS, the Story scope is set to *Grouped*, a story in this context means the contiguous collection of GUI rows with the same story name. See Story Scope, in section *Settings Editor MOS*.

Story items that follow can be programmed to inherit a given newsroom tag, with an optional default value.

Keep Newsroom tag (with the same name) within a story

If a newsroom tag has the special value *KEEP*, this value will act as a placeholder in the sense that the tag will get the value of the last newsroom tag in the story that has the same name.

> **Note:** The special value KEEP is configurable . See definition for **Newsroomtag keep keywords** in Settings Editor MOS.

Inherit from a newsroom tag with another name

A newsroom tag may *inherit* from another tag *in the same story*. In this case, the inheriting tag gets the same value as the value of the referred tag.

This is done in the following way: The value of one tag (the *inheriting tag*) may be given as *{name}*, where *name* is the name of referring tag.

Example: If there is a tag with name *left*, another tag (with another name) may inherit the tag *left* by having the value *{left}*.

Inherit with default value

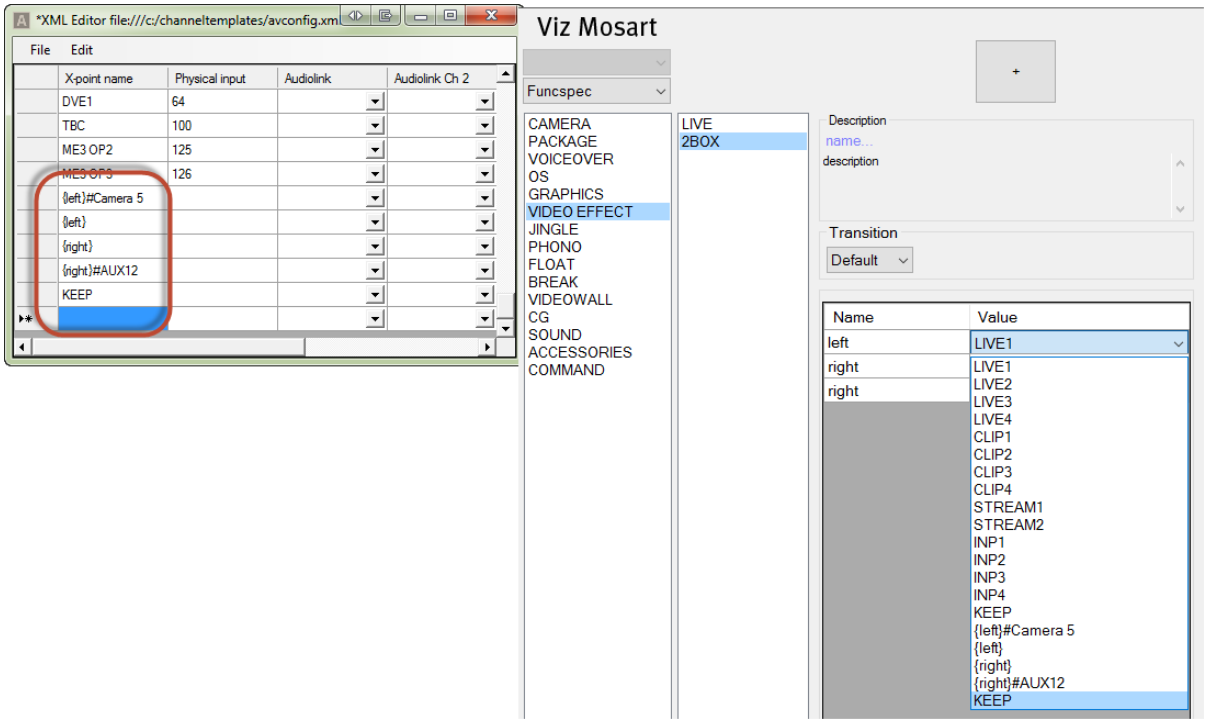
Optionally, a *default* value may be given, to be used when the tag inherited from within the story, has no value or is undefined.
The syntax for the value of the inheriting tag is *{name}#default*, where *name* is the tag that the value is inherited from and *default* is the default value.

Example: Continuing the example above, the inheriting tag may be given the value *{left}#OS 2*. Then, if the tag *left* has no value, or does not exist, the value of the inheriting tag will be set to *OS 2*.

Adding inheriting values to the lists defined in AV Automation's A/V Setup

A newsroom tag can be listed among the definitions in AV Automation's **A/V Setup** menu.
You can add the special value *KEEP*, *{name}*, and *{name}#default* to these lists, making them available for selection both in Template Editor and in the ActiveX.

In the example below, five of these values have been added to the video configuration list. To the right, how these values appear in the ActiveX:



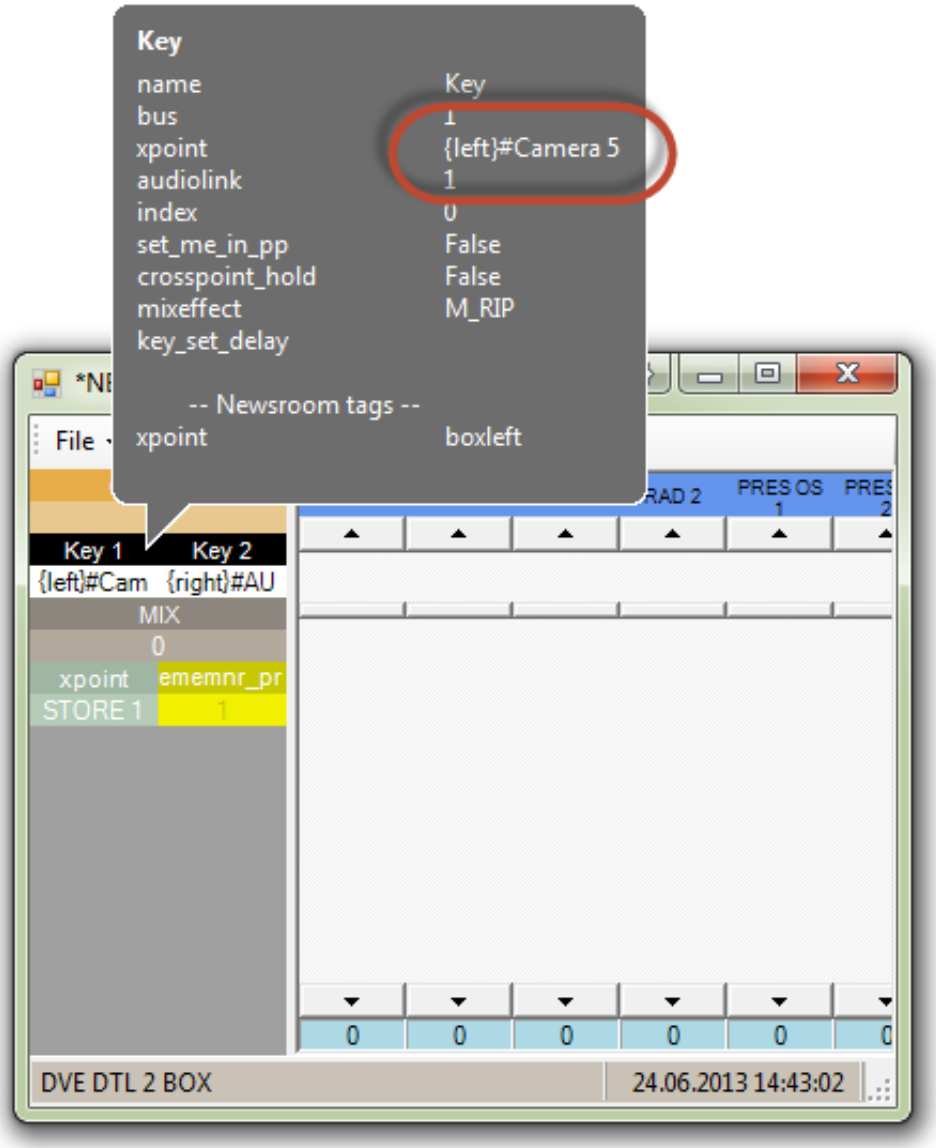
Explanation:

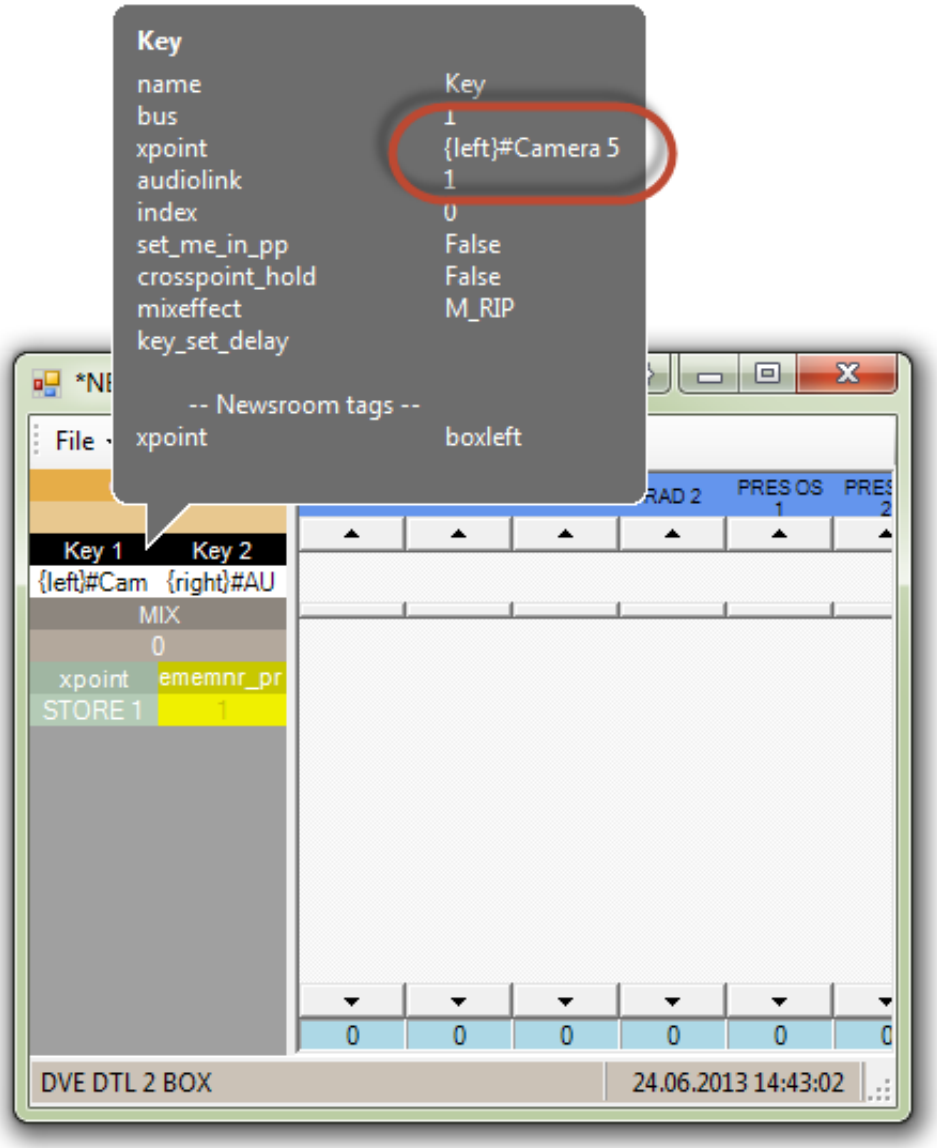
- {left}#Camera 5: Use the value of the last Newsroom tag {left} or *Camera 5* if no such newsroom tag exists.
- {left}: Use the value of the last Newsroom tag {left}. Nothing happens if no such Newsroom tag exists
- KEEP: Special configurable word that implies retaining the existing value of the corresponding newsroom tag. (In the ActiveX example, *left*)

Examples

DVE Template with inheritance

A DVE 2BOX template where the left and right input may be inherited from sources within the same story.


Key
name Key
bus 1
xpoint {left}#Camera 5
audiolink 1
index 0
set_me_in_pp False
crosspoint_hold False
mixeffect M_RIP
key_set_delay
-- Newsroom tags --
xpoint boxleft
File
Key 1
Key 2
{left}#Cam
{right}#AU
MIX
0
xpoint
ememnr_pr
STORE 1
1
PRES OS 1
0
DVE DTL 2 BOX
24.06.2013 14:43:02



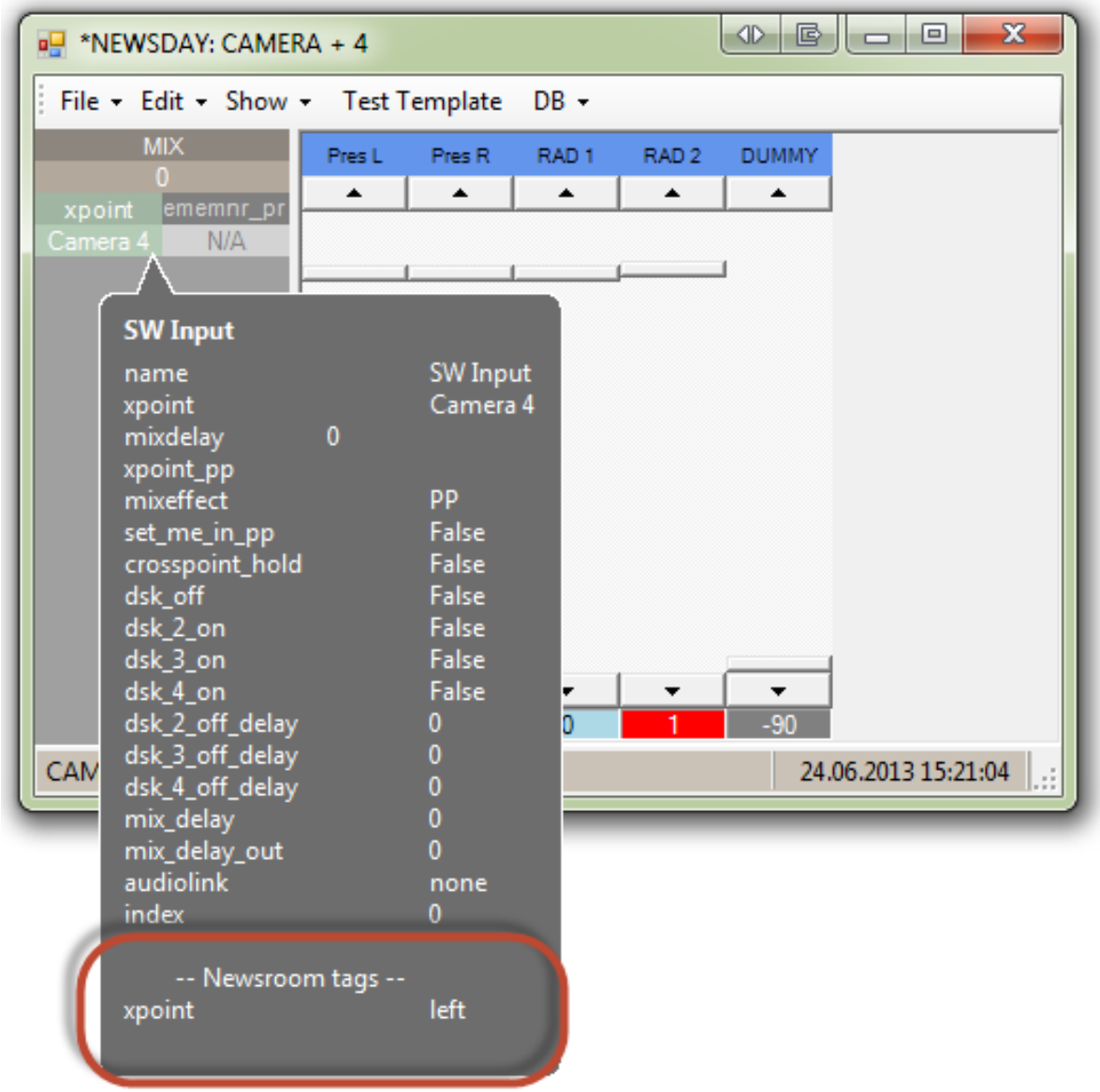
In this template the values for Key 1 and Key 2 are:
Key 1 = {left}#Camera 5
Key 2 = {right}#AUX 12
Key 1 will then be either the current value of a newsroom tag named *left* in the same story, or *Camera 5* if no such newsroom tag exists.

Timeline Example

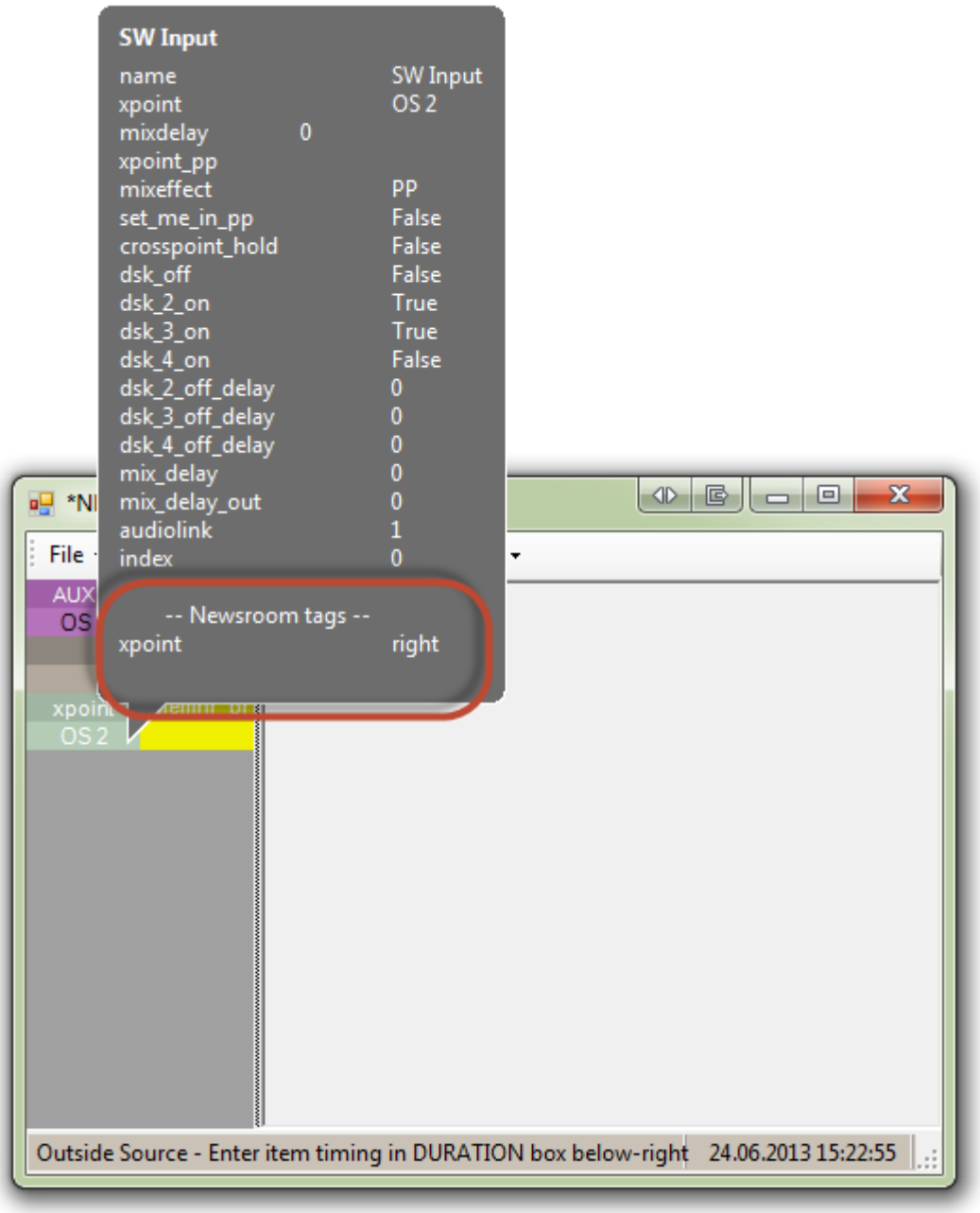
Camera+4: left = Camera 4	**LIVE+2:** right = OS 2	**DVE+DTL2BOX:** left = {left} right={right}

In the figure above, three templates are placed in a timeline:

1. Camera+4: Where xpoint is set to *Camera 4* with Newsroom tag = *left*.

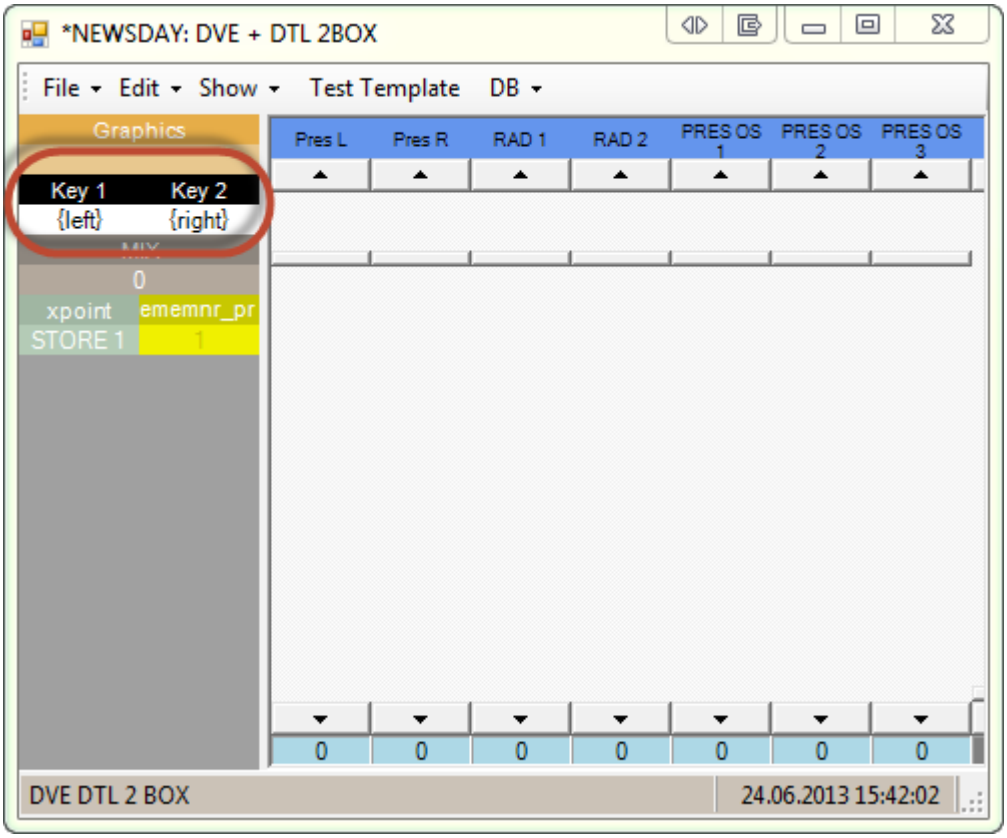


2. Live+2: Where xpoint is assigned to OS 2 with Newsroom tag = *right*.



Note: The newsroom tag is entered here to allow later templates in the story to inherit the value. Adding a newsroom tag here also exposes this value to the ActiveX.

3. DVE+DTL2BOX where left and right inputs are set to inherit newsroom tag *{left}* and *{right}* respectively.



In this example, the *left* and *right* inputs are inherited from previous templates in the same story, exposing newsroom tags with the same names.

Note: Here, *no* default values are given. No action is taken if no values are assigned to the *left* and *right* newsroom tags.

9.3.5 Control Commands in Templates

A Mosart *control command* can be triggered from a template, executed when the template is cued (*Commands On Cue*), taken (*Command On Take*), at a continue point (Continue Points) or when the template is taken out (*Command On Take Out*).

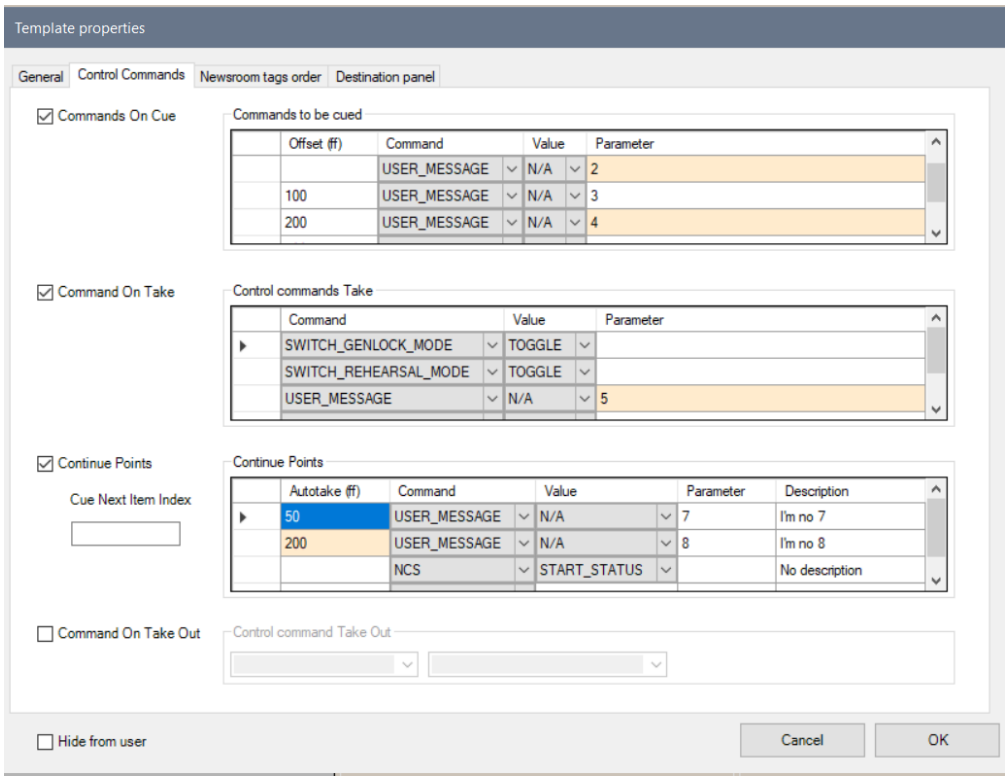
- Working with Control Commands
- Command Values and Parameters
- VIDEO_PORT Control Commands in AV Automation Template Properties
- RECORD Control Commands in AV Automation Template Properties

Info:

- A Mosart template can trigger *multiple* control commands.
- A (not-templated) control command can trigger a Mosart template, often used for a *DirectTake* template.

Working with Control Commands

This is done through **AV Automation > Devices > Template Editor > Template > Properties > Control Commands** tab.



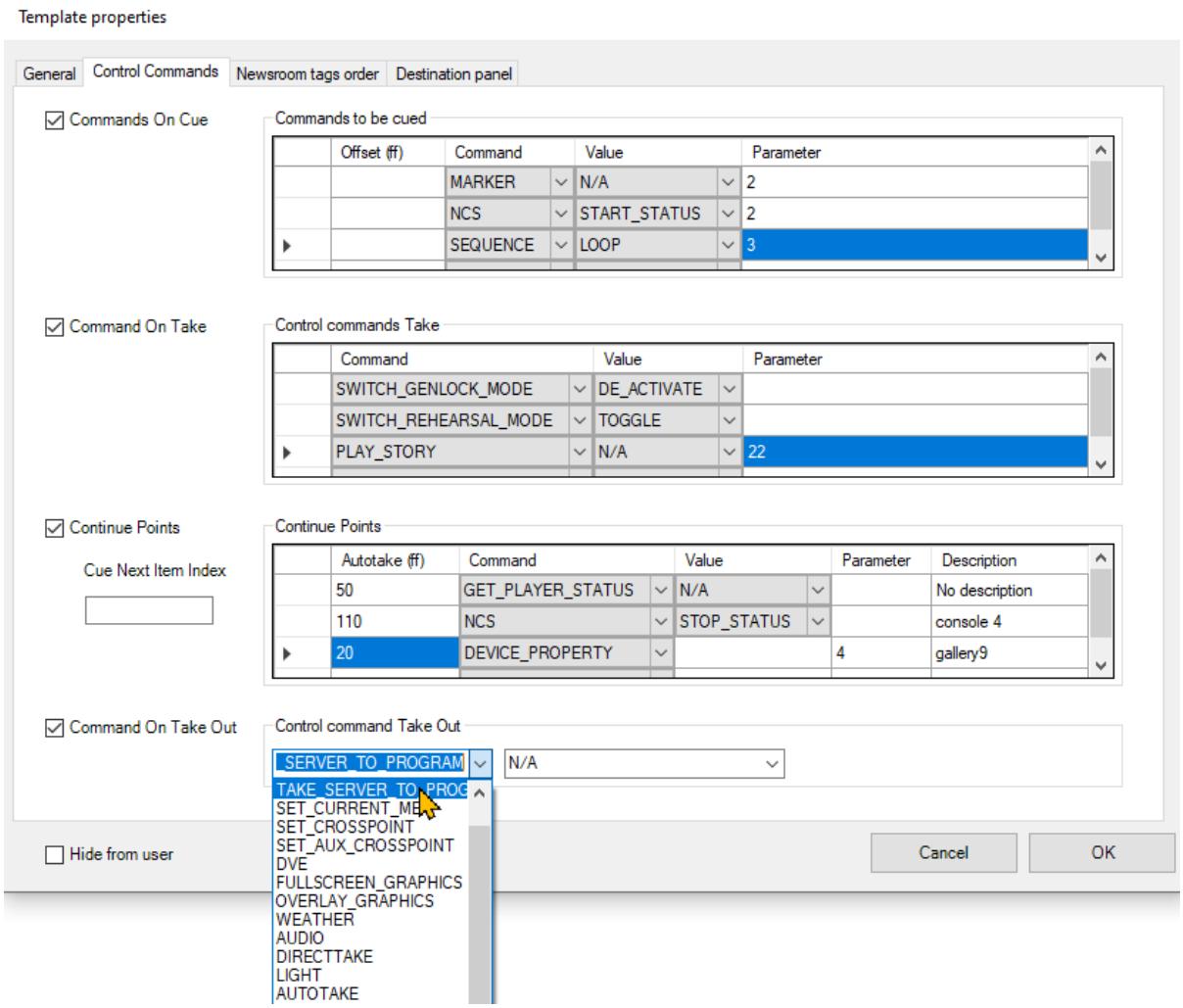
The figure above shows a template with several control commands Executed when the template is *cued*, when it is *taken*, and at Continue points while the template is on air.

Control commands can also be executed when the template is *taken out*. Note that *Parameter* settings are *not* available at Take Out. In this context, triggering a Direct Take may be more appropriate.

Adding a Control Command

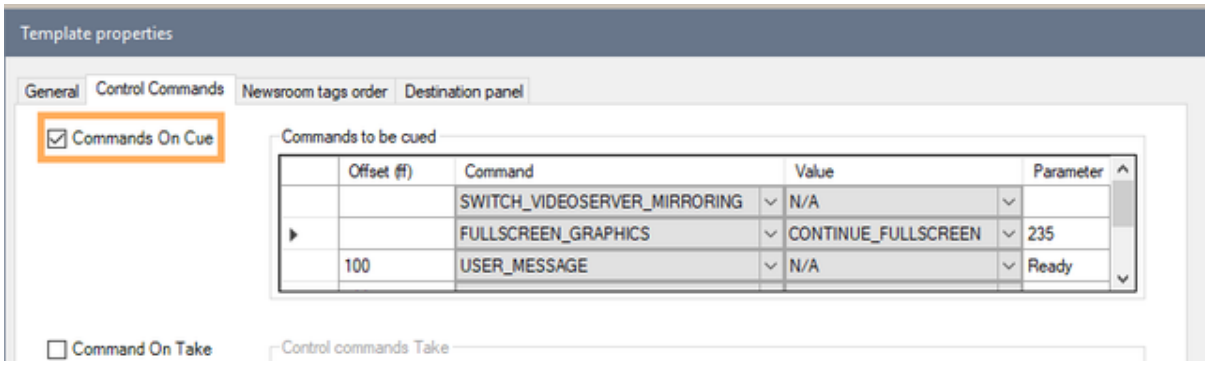
To add a control command

1. In the **Template properties** window, click the **Control Commands** tab.
   There are four categories of control command:
   a. Commands On Cue
   b. Command On Take
   c. Continue Points
   d. Command On Take Out.
2. Refer to the table Command Values and Parameters below for details.



Commands On Cue

You can recall control commands in Preview, when the template is cued. This enables triggering of events when a story item is taken to Preview.



As illustrated above, you can give the command a *Value* and a *Parameter*. Refer to the Command Values and Parameters below for which values and parameters are applicable for a given command.
Also, in a similar way as *Autotake* offsets for Continue Points, you can set a frame number value of **Offset** governing when the command is to be executed.

> **Info**: For *Cue* commands, all offsets are relative to the *start of cueing*. This differs from *Continue Points*, where each offset is relative to the *previous* continue point.
> In the illustration above, the User Message (*Ready*) will be sent 100 frames after cueing. Commands without offset will be executed immediately On Cue.

By defining a *Newsroom tag* as a parameter for an *On Cue* command, you can set the parameter from the NRCS. For *Commands On Cue*, this applies to the *Parameter* column.

Parameters as Newsroom Tags

- In the **Parameter** column, right-click (once or twice) and select either *Add*, *Remove* or *Change* the newsroom tag.

> **Tip**: The selection with all parameters appear in the Mosart Web App *NRCS Plugin*, simplifying the task of adding the template to a rundown.

- The **Parameter** field gets a pink background color, indicating that a Newsroom tag is present.
- Hovering-over the field displays the value of the Newsroom tag.

Command On Take

When selected, the commands inserted in **Command On Take** are executed when the template is taken to Program or on-air.

You can give the command a *value* and a *parameter*.
Refer to Command Values and Parameters below for applicable values and parameters for a given command.

By defining a *Newsroom* tag as a parameter for an *On Take* command, the user can set the parameter from the NRCS.
For *Command On Take*, this applies to the **Parameter** column.

- In the **Parameter** column, right-click (once or twice) and select either *Add*, *Remove* or *Change* the newsroom tag.

> **Tip**: This selection then appears in the Mosart Web App *NRCS Plugin* for adding the template to a rundown.

- The **Parameter** field gets a pink background color, indicating that a Newsroom tag is present.
- Hovering-over the field displays the value of the Newsroom tag.

Continue Points

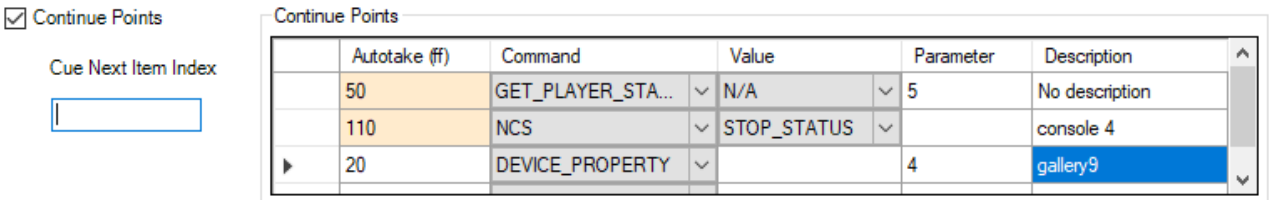
A Continue point may be executed in one of two ways:

- The user presses TAKE NEXT (default: **F12**).
- If a number is entered in the **Autotake** column (see below), the Continue point will be executed that many frames after the previous Continue point was executed (or, if this is the first Continue point, from when the template was taken on air).

To enable Continue points

- To enable this functionality, tick the **Continue Points** checkbox.

The Continue points are defined in this table:



As illustrated above, you can give the command a **Value**, **Parameter** and **Description**. Refer to section Command Values and Parameters about which values and parameters are applicable for a given command.
Description is helpful for explaining the purpose of this command.

- **Cue Next Item Index**: Enter the line number (for example, *0, 1, 2, 3*) of the Continue point that executes a cue for the next/following Primary object. If empty, Viz Mosart cues when entering the last continue point in the list. In other words:
  a. if **Cue Next Item Index** is *empty*, then Mosart will cue the next item in the rundown after all the continue points have been taken to air.
  b. if **Cue Next Item Index** is *0*, then Mosart will cue the next item immediately after this item has been taken.
  c. if **Cue Next Item Index** > *0*, then Mosart will cue the next item after the continue point with the number specified by the value of **Cue Next Item Index** field has been taken to air.

- **Autotake** (optional): The autotake time, in frames. The command is then automatically taken after that time (unless taken manually earlier by the user pressing TAKE NEXT). If empty, the command *must* be executed manually by the user pressing TAKE NEXT.

> **Info**: For *Continue points,* all autotake offsets are relative to the last Continue point (or, for the *first* Continue point, to the time the template was taken). This differs from *Commands on Cue*, where each offset is relative to the *start of cueing*.
> The order that the instructions appear in the table is important. Looking at the example above:
> 1. The first (Player Status) will be run 50 frames after Take
> 2. The next one (status from the NRCS) 110 frames later (160 frames after Take) and
> 3. The third 20 frames later (180 frames after Take).

By defining a *newsroom* tag as a parameter for **Autotake**, you can set the parameter from the NRCS.

- In the **Autotake** column, right-click and select either *Add*, *Remove* or *Change* newsroom tag.

> **Tip**: This selection then appears in the Mosart Web App *NRCS Plugin* for adding the template to a rundown.

Note also that the *Autotake* field will get a pink background color to indicate that a Newsroom tag is present. Hovering over the field displays the Newsroom tag value.

Command On Take Out

The commands inserted here are executed when the template is taken from Program, taken off air, or at the end of the template. Only a single command can be used in **Command On Take Out**, and *no* parameters are accepted.

You can give the command a *Value*. Refer to Command Values and Parameters below for applicable values to a given command.

Command Values and Parameters

> **Info:** Commands where the Parameter field is marked with (*) do not work for templates.

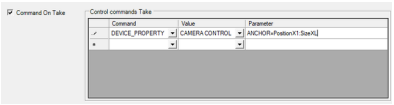
> **Note:** It is not possible to enter a *Parameter* in the **Command On Take Out** section.

COMMAND	Value	Parameter	Description
ACCESSORIES	TAKE_NEXT	(*)	

COMMAND	Value	Parameter	Description
AUDIO	FADE_MANUAL		Toggle fade manual.
	FADE_OUT_KEEPS	Fader rate	Faders that are set as *keep level* in the template On Air are faded out. Will also work when new templates have been taken in between the *keep* action and the FADE_OUT_KEEPS.
	FADE_DOWN SECONDARY_AUDIO	Fader rate	Viz Mosart fades down relevant audio sources not connected to the video currently on air.
	FADE_UP SECONDARY_AUDIO	Fader rate	Viz Mosart fades up relevant audio sources not connected to the video currently On Air, when cutting between sources.
	SET_LEVEL_2_ONAIR	TRUE FALSE <empty>	These settings apply to fader second levels in the template that is *On Air*. TRUE: The second level for faders are set to *ON*. FALSE: The second level for faders are set to *OFF*. <empty>: (no value entered) The second level for faders are set to *TOGGLE their current state* (default). **Info**: A blank command parameter is a toggle function, and is therefore *not* suited for use from a template.

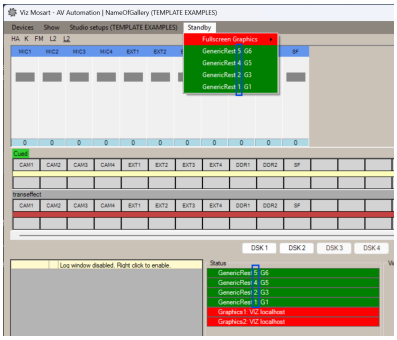
COMMAND	Value	Parameter	Description
	SET_LEVEL_2_PREVIEW	<empty>	This setting applies to fader second levels in the template that is in *Preview.* The operation will be performed at next transition. <empty>: (no value entered) The second level for faders in the template in Preview are set to *TOGGLE their current state.* **Info**: A blank command parameter is a toggle function, and is therefore *not* suited for use from a template.
	FREEZE_AUDIO	TRUE FALSE <empty>	TRUE: All sound faders are frozen, and do not respond to commands from templates. FALSE: All sound faders respond to commands from templates. <empty>: (no value entered) All faders are set to *TOGGLE their current response state* (default). **Info**: A blank command parameter is a toggle function, and is therefore *not* suited for use from a template.
AUTOTAKE	N/A	(*)	

COMMAND	Value	Parameter	Description
AUTOTRANS	PP, ME1, ME2, ME3, ME4	MixEffect (MIX OR WIPE) + Transitionrate	For example: *MIX 33, WIPE 10* **Info**: The *Effect* cannot be used.
	AUX, Default	(*)	
CROSSOVERCOMMAND	N/A	(*)	

COMMAND	Value	Parameter	Description
DEVICE_PROPERTY	CAMERA CONTROL	See *Description* column to the right.	**Info**: This is for *camera robotics*. When the DEVICE_PROPERTY control command is used with the value *CAMERA CONTROL*, only setting speed (Camerobot) and *anchor* (Camerobot/FxMotion) are controllable from a template. Details for Camerobot **ANCHOR**: ANCHOR=pos1:presenter1[,pos2:presenter2]... The position and presenter is given as a pair separated with ':'. Multiple position-presenter pairs are possible, separated by ','.  **Info:** The position is a *string* and does not have to be a number. Details for Camerobot **Time**: If Time =1 (>0), Viz Mosart sends MATRIX_MOVE_MPOS command to Camerobot, while if Time = 0 Viz Mosart sends MATRIX_CUT_MPOS command Camerobot.

COMMAND	Value	Parameter	Description
	AUDIO	Requires <fields> structure as parameter with parameter1=AUDIO, parameter2=key, parameter3=value or connectionstring style	**Info**: This is for an *audio mixer*.
DEVICE_STANDBY	AUDIO_MIXER, GPI, LIGHTS, LOUDNESS, ROUTER, SOUND_PLAYER, SUBTITLING, VIDEO_SERVER, VIDEO_SWITCHER, VIDEO_WALL, VIRTUAL_SET, WEATHER	ON OFF TOGGLE	ON: The device chosen (in Value) is put in Standby. OFF: The device chosen (in Value) is taken out of Standby. TOGGLE: The Standby status (ON/OFF) of the device chosen (in Value) is toggled. (If there are several devices of the chosen kind (e.g. video servers), *all* of them are affected.)
DEVICE_STANDBY	GRAPHICS (full-screen)	As above (ON/OFF/TOGGLE) or engineno:[ON|OFF|TOGGLE] (Example: *2:TOGGLE*)	ON/OFF/TOGGLE: As above, applied to all graphics engines. If an engine number is given, only the designated engine is affected.
DEVICE_STANDBY	ROBOTIC_CAMERA	As above (ON/OFF/TOGGLE) or controller:[ON|OFF|TOGGLE] (Example: *1:ON*) or controller:camera:[ON|OFF|TOGGLE] (Example: *1:2:ON*)	ON/OFF/TOGGLE: As above, applied to all cameras on all controllers. If a controller (but no camera) is given, only the cameras on the designated controller are affected. If both a controller and a camera is given, only the designated camera is affected.

COMMAND	Value	Parameter	Description
DEVICE_STANDBY	OVERLAY_GRAPHICS	As above (ON/OFF/TOGGLE) OR Engine No:[ON|OFF|TOGGLE] (Example: *2:TOGGLE*: Toggles Engine 2 Standby status) OR Destination name: [ON|OFF|TOGGLE] (Example: *DSK:TOGGLE*: Toggles Destination DSK Standby status)	ON/OFF/TOGGLE: Applied to all overlay graphics (all controllers and corresponding engines). If an engine number or destination name is given, only the designated engine or destination is affected. If the given number is both engine number of a overlay graphic and the destination name of some overlay graphics, all those overlay graphics are affected. **Info**: Engine numbers are shown in the list of active engines in Overlay Graphics Interface and followed by '/' and destination name of the engine. (*Engine Handler*, *Destinations*, *Controller* have *no* effect).

COMMAND	Value	Parameter	Description
DEVICE_STANDBY	GENERIC_REST	As above (ON/OFF/TOGGLE) OR GenericRest No:[ON|OFF|TOGGLE] For example: *1:TOGGLE*: Toggles GenericRest 1 Standby status)	ON/OFF/TOGGLE: Applied to all generic REST devices. If a generic REST number is given, only the designated generic REST device is affected. The generic REST number can be found in AV Automation Status panel and Standby list. 
DIRECTTAKE	The Recall Nr (see Template Properties) of the direct take template to execute	-	Executes the direct take template provided in **Value**.
DVE	FORWARD REVERSE	Recall forward/ reverse EMEM	n/a
ENABLE_GRAPHICS_MIRRORING	n/a	(*)	

COMMAND	Value	Parameter	Description
FULLSCREEN_GRAPHICS	CONTINUE_FULLSCREEN	AUTO	Continue Command Only: The AUTO parameter will replace the continue command with the number of continue points in the graphics item. The number of continue points is taken from the graphics_continuecount field. Examples: *AUTO+1*: The number of continue counts from the graphics + 1 *AUTO-2*: The number of continue counts from the graphics - 2
FULLSCREEN_GRAPHICS	CONTINUE_FULLSCREEN	Engine no	The target output fullscreen engine number to execute the action on (for example, *1,2,3*).
	MACRO	Engine no : macro	The target output fullscreen engine number to execute the action on (for example, *1,2,3*) and Macro name. The engine number must be same as the number defined in AV Automation
GRAPHICSPROFILE	N/A	-	
HOLD_AUDIO_TRANSITION	N/A	-	
HOLD_VIDEO_TRANSITION	N/A	-	

COMMAND	Value	Parameter	Description
ITEM_UPDATE_FIELDS	N/A	Must be a comma separated list in the format: Bus,Newsroomtag,Crosspoint Example: *Preview,left,CAM1*	Update the source (Crosspoint) assigned to a Newsroom tag, for the item in PREVIEW or in PROGRAM. **Bus**: Preview or Program **Newsroomtag**: Newsroom tag name that will be updated. **Crosspoint**: Video crosspoint name in AV Config.
LIGHT	N/A	(*)	
MARKER	N/A	(*)	
NCS	START_STATUS, STOP_STATUS	RUNDOWN STORY ITEM	Specific for Open Media
NEXT_CUE_DELAY	EXTEND	A number of frames	If there is a pending cue (when this control command is executed), this cue is *postponed* by the number of frames given. Otherwise (no pending cue), the next cue operation will be *additionally delayed* by the number of frames given. (I.e., the value will be added to the sum of the settings • Min. Cue delay in AV Automation Devices Vision Mixer • Next cue delay in Audio and Video Setup, Vision Mixer Effects Configurations, Settings (if an effect is involved).

COMMAND	Value	Parameter	Description
OVERLAY_GRAPHICS	CLEAR	Destination Current Render	From where to clear which graphics. **Destination**: Non-numerical value. **Current**: Clear all overlay graphics currently on air. **Render**: Numerical value, the Engine number.
	CONTINUE	Destination or graphics ID	The parameter can be either one of the Mosart graphics destinations (channels) or a valid graphics ID.

COMMAND	Value	Parameter	Description
	MACRO	Engine no:macro or Destination:macro	The *engine number* can be identified from the Overlay Graphics Interface (OGI) screen, where device statuses are displayed. For Vizrt Graphics specifically, the *destination* configured in OGI can also be used in the command. Examples: *4:macrotest* *DSK:macrotest* When sending a macro command for Vizrt Graphics, Mosart will direct it to the Media Sequencer (MSE), specifying the destination linked to the engine number provided in the command. In cases where multiple engines are mapped to the same destination (commonly seen in a mirroring setup), the Media Sequencer (MSE) will send the command to all engines associated with that destination, not just to the engine with the number specified in the command. **Info:** For Vizrt Graphics, when multiple controllers are defined in OGI, the engine number shown in the OGI screen is not reliable and instead, the *destination* should be used.
	PRETAKE_NEXT_OVERLAY	Destination	Destination (handler) to pretake the next overlay on.

COMMAND	Value	Parameter	Description
	TAKE_LAST_OUT	(*)	No function from templates. Can only be used from keyboard shortcut or remote control command.
	TAKE_NAMED_OVERLAY	Name/slug of the named overlay to take	
	TAKE_MANUAL_OUT	Destination or Empty	Destination to take out the manual graphics from. Non-numerical value: Destination Empty or Numerical value: Takes out all the manual graphics **Note:** For versions before Viz Mosart 5.5.0, the TAKE MANUAL OUT command would also take out overlay graphics set to wait for background end or story end. If this functionality is needed, use the below TAKE ALL OUT command introduced in Viz Mosart 5.5.0.
	TAKE_ALL_OUT	Destination or Empty	Destination to take out the graphics from. Non-numerical value: Destination Empty or Numerical value: Takes out all the graphics

COMMAND	Value	Parameter	Description
OVERLAY_TO_MANUAL	Parameter 1: ONAIR (default), PREVIEW.	Parameter 2: Comma-separated list of handler names Parameter 3: AUTOMATIC (default) / MANUAL Parameter 2 and Parameter 3 must be separated by a semicolon. (E.g. WALL,DSK;MANUAL.)	**Info:** If there is no semicolon after Parameter 2; nothing to the right of the semicolon; or Parameter 3 has any value other than 'MANUAL', then Parameter 3 gets the default value 'AUTOMATIC'. • For user operations, see the Viz Mosart User Guide, section *Viz Mosart User Interface > Keyboard Shortcuts > Control Command Keys > OVERLAY_TO_MANUAL*.
PLAY_STORY	N/A	(*)	
RECORD	PREPARE, START, STOP	Split parameters using, - first: clipname, - second: recorder (optional), - third: port name (default *Rec*), - fourth: group name (default *Rec*)	For example: *clip33,recorder1,Rec,Rec* • For more configuration details, see the section RECORD Control Commands in AV Automation Template Properties below.
RELEASE_BACKGROUND	N/A	(*)	
RESET_OVERUNDER	N/A		Resets the Rundown Over/Under timer to *00:00*.

COMMAND	Value	Parameter	Description
RUNDOWN_NCS_RESYNC	N/A	(*)	
SEQUENCE	LOOP	-	Sets a running sequence to start looping
	STOP_LOOP	-	Stops running a looped sequence in loop
	STOP	-	Stops a running sequence
	START	-	Restarts a previously stopped sequence
SET_AUX_CROSSPOINT	N/A	(*)	

COMMAND	Value	Parameter	Description
SET_CROSSPOINT	PP, ME1, ME2, ME3, ME4 AUX Default	bus/layer:xpoint	Two arguments, separated by a colon [:], are supported. The first parameter is the *Mixer Region's bus*. The second is the *crosspoint*, as configured in A/V Setup. **Info**: • If no colon is used, the value is interpreted as only a crosspoint, and bus *A* is used. • For the TriCaster video switcher, it is possible to specify the layer (*A,B,C,D*) in addition to the crosspoint, enabling control of the buses for M/E effects with 4 layers. **Examples**: *PP CAM1* - Switch source CAM1 in Program *PP A:CAM1* - Same as above *PP B:CAM1* - Switch source CAM1 in Preview *ME1 CAM2* - Switch source CAM2 in Program *ME1 B:CAM2* - Switch source CAM2 in Preview *ME1 1:CAM1* - Switch source CAM1 for keyer 1 *ME1 2:CAM2* - Switch source CAM2 for keyer 2 *ME1 C:CAM1* (TriCaster) - Switch source CAM1 for the the third layer of M/E effect.

COMMAND	Value	Parameter	Description
SET_CURRENT_ME	PP, ME1, ME2, ME3, ME4 AUX Default	(*)	
SET_VIDEOSERVER_SALVO	Select a configured video server from the list	-	These same settings are available as keyboard shortcuts, as presented in the *Viz Mosart User Guide*, under section *Viz Mosart User Interface*.
STUDIOSETTUP	#	-	
SWITCH_GRAPHICS_MIRRORING	TOGGLE, ACTIVATE, DE-ACTIVATE	-	For standard and Story Recorder (SR) shows.
SWITCH_GENLOCK_MODE	TOGGLE, ACTIVATE, DE-ACTIVATE		Story Recorder (SR) specific: As long as some standard show elements are not supported with SR, use this command to control the use of Genlock mode. SR can only run in Genlock mode.
SWITCH_REHEARSAL_MODE	TOGGLE, ACTIVATE, DE-ACTIVATE		Story Recorder (SR) specific: SR can only run with Rehearsal mode deactivated, use this command to ensure Rehearsal mode is not accidentally active.
SWITCH_VIDEOSERVER_MIRRORING	N/A	-	For standard and Story Recorder (SR) shows. This command has no parameters, it simply acts as a toggle.

COMMAND	Value	Parameter	Description
TAKE_CONTINUE_POINT	N/A	(*)	
TAKE_SERVER_TO_ PROGRAM (as a Continue Points command)	Transition Rate (frames)		Takes a video server port to program. For example: a video clip is running on a video wall. The control command can then be used to take the last used/active video server to program. The transition will be done over the number of frames given as Value to this command.
TAKE_SERVER_TO_ PROGRAM (as Commands On Cue, Command On Take or Command On Take Out)	N/A		Takes a video server port to program. For example: a video clip is running on a video wall. The control command can then be used to take the last used/active video server to program. This will be done as a cut, since it is not allowed to define a transition rate for these cases.
TRANSITION_TYPE	CUT, MIX	Rate	
	EFFECT	Effect no	
	TOGGLE	(*)	
TRIGGER_START	N/A		Plays the cued clip from the taken template that had TRIGGER START enabled in the video server device, see Template Device Functions#Video-Server .
UNKNOWN	N/A		Default fallback. Does nothing.

COMMAND	Value	Parameter	Description
VIDEO_PORT	N/A	Must be a comma separated list in the format: command,port, parameter	For example: PLAY_PAUSE,A,PLAY. • For more configuration details, see the section VIDEO_PORT Control Commands in AV Automation Template Properties below. • For user operations, see Viz Mosart User Guide, section *Viz Mosart User Interface > Keyboard Shortcuts > Video Port Control Commands >Video Port Keyboard Shortcuts*.
VIDEO_SERVER_GOTO	N/A	(*)	
VIDEOWALLMODE	N/A	(*)	
WEATHER	N/A	(*)	

Parameters with Placeholders

Control command parameters can include *placeholders* which are then populated with values found among the fields of a currently on air, Viz Mosart item. A placeholder is defined as a string of characters, within curly brackets {placeholder}.
For example, a placeholder can be useful for transmitting values from the NRCS, for use as control commands that will be executed when the template is taken on air.

Example: Providing the clip name to be recorded, from the NRCS.

In the NRCS, a column can be created for giving the name of the clip to be recorded with Viz Mosart. Then, in Viz Mosart Newsroom settings, map the column in *Story External Metadata* as follows:

Story External Metadata

```
<mostag mostagname="NCScolumn" mosart_action="item_variable"
 action_value="NCS_filename" format_type="String" />
```

In Manus file, the Mosart item will have the following fields (the value *Clip1* is given from the NRCS):

Mosart Item

```
<fields>
     <field name="NCS_filename" value="Clip1" />
</fields>
```

Then, in **Template Editor**, the control command for the respective template is configured as follows:

Control Command

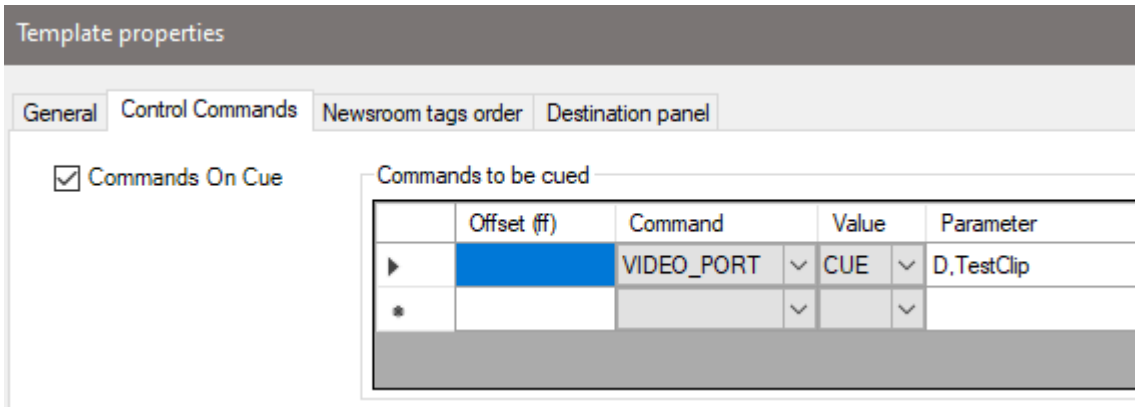
```
RECORD PREPARE {NCS_filename}_Test,Recorder,RecPort
```

When the template is executed, a clip with name *Clip1_Test* is prepared for recording.

VIDEO_PORT Control Commands in AV Automation Template Properties

- VIDEO_PORT: Syntax
- VIDEO_PORT: Commands
- VIDEO_PORT VideoPort Syntax

The VIDEO_PORT control commands can also be used in templates. The syntax order of the parameter is important.



Examples

Examples using port A:

```
PLAY_PAUSE,A,PLAY PLAY_PAUSE,A,PAUSE PLAY_PAUSE,A
STOP,A STOP,A,CUE
RECUE,A,PLAY RECUA,A
SET_LOOP,A SET_LOOP,A,OFF SET_LOOP,A,12-444
CUE_TAIL,A,-15 CUE_TAIL,A,15
PLAY_TAIL,A,-15 PLAY_TAIL,A,15
```

VIDEO_PORT: Syntax

Value=Command, Parameter=[@Delay,]VideoPort,Properties

VIDEO_PORT: Commands

- PLAY_PAUSE: Viz Mosart will begin to play the cued content. If the content is already playing, it will pause the content
    - No Properties: PAUSE if playing and PLAY if paused
    - Properties=PLAY: Always try to PLAY channel
    - Properties=PAUSE: Always try to PAUSE channel.
- STOP: Viz Mosart will stop the cued content
    - Properties: CUE. Cue the channel after the stop.
- CUE: Viz Mosart will initially CUE the port
    - Properties: ClipName.
- RECUE: Viz Mosart will RECUE the port
    - Properties: PLAY. Viz Mosart will play the clip after the recue.
- SET_LOOP: Viz Mosart will set the already cued clip to looping
    - Properties: OFF, RESET. Viz Mosart will turn the looping off
    - Properties: 12-444 (configurable). Viz Mosart will set the looping from frame 12 to frame 444.
- CUE_TAIL: Viz Mosart will recue the cued clip at a specific time (in seconds)
    - Properties: -15 (negative value) - configurable. Viz Mosart will cue at the start of the last 15 seconds of the clip (the clip will have 15 seconds left to play plus Post roll)
    - Properties: 15 (positive value) - configurable. Viz Mosart will cue 15 seconds into the clip.
- PLAY_TAIL: Viz Mosart will recue and play the cued clip at a specific time (in seconds)
    - Properties: -15 (negative value) - configurable. Viz Mosart will cue at the start of the last 15 seconds of the clip
    - Properties: 15 (positive value) - configurable. Viz Mosart will cue 15 seconds into the clip.

All commands may be prefixed with an optional delay given in frames. If a delay is specified then the Parameter section shall start with an '@' character.

VIDEO_PORT VideoPort Syntax

The VideoPort may be given either explicitly or by using one of the following identifiers:

- ONAIR: The current on-air video port used by a primary story item
- PREVIEW: The current video port in preview from a primary story item
- VideoPortGroup,CURRENT: The current video port for the given video port group.

VIDEO_PORT Examples:

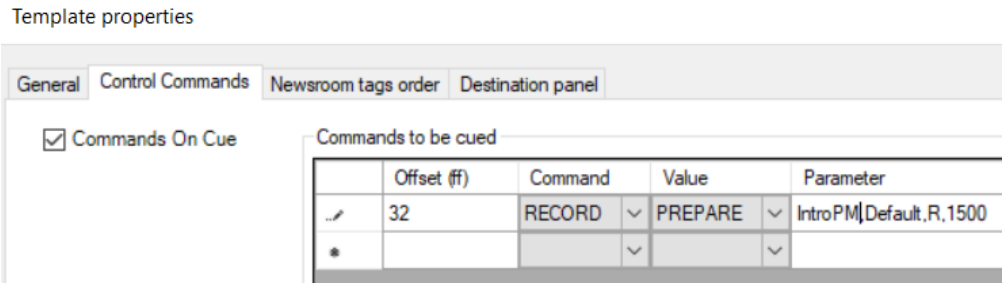
Value	Parameter / Properties	Description
PLAY_PAUSE	D	Alternatively plays / pauses the clip cued in Mosart port D
PLAY_PAUSE	D,PLAY	Always do a play for the clip cued in port D

Value	Parameter / Properties	Description
PLAY_PAUSE	D,PAUSE	Always do a pause for the clip cued in port D
STOP	D	Stops any running clip in port D
STOP	D,CUE	Stops any running clip in port D followed by a recue
CUE	D,TestClip	Cues the clip TestClip in port D.
CUE	@25,D,TestClip	Cues the clip TestClip in port D after a delay of 25 frames
RECUE	D	Recues the clip in port D
RECUE	D,PLAY	Recues the clip in port D and plays it

RECORD Control Commands in AV Automation Template Properties

- Examples
- RECORD: Syntax

The RECORD control commands can also be used in templates.



Examples

Value	Parameter	Description
PREPARE	TestClip,Default,R	Prepare a clip with name *TestClip* on the *Default* recorder. The *Default* recorder will be using video port *R*. For Quantel this clip will be initiated using the default duration.
PREPARE	TestClip,Default,R,1500	Prepare a clip with name *TestClip* on the *Default* recorder. The *Default* recorder will be using video port *R*. For Quantel this clip will have an initial duration of 1500 frames = 1 minute (25 frames/sec)
PREPARE	TestClip;Category=Totem ,Default,R,1500	Same as above but the clip will be assigned to the “Totem” category in a Quantel video server system. **Info**: There is a semicolon between *TestClip* and *Category*.
START	,Default	Starts the recording on the *Default* recorder. **Info**: There is a leading ‘,’ in the *Parameter*
STOP	,Default	Stops the recording on the *Default* recorder **Info**: There is a leading ‘,’ in the *Parameter*

The options in the *Value* dropdown are

- PREPARE
- START
- STOP
- STOP_AND_DETACH
- DELETE.

The *Parameter* should be a comma-separated list of up to four items:

- *ClipName*
- *Recorder*
- *PortName*
- *Duration*.

If all four items are present, the Parameter will be

<ClipName>,<Recorder>,<PortName>,<Duration>

RECORD: Syntax

If not all items are present, any commas to the left of present items will still have to be present.
For example, if only *Recorder* is present, the syntax is:

,*<Recorder>*

In some cases, the *ClipName* item may have sub-items. In that case, the sub-items are separated by semicolons. (An example is given below.)

Here the options in the *Value* dropdown are explained in more detail:

- PREPARE: Viz Mosart will prepare a clip for recording.
  - *Parameter: <ClipName>,<Recorder>,<PortName>,<Duration>*
  - *ClipName*: The name of the clip to be created
  - *PortName*: The name of the video port where the recording shall take place. Note that this port has to be part of the '*Rec*' virtual video port group.
  - *Recorder*: A user-defined name to be used for the recording, tied to *ClipName* and *PortName*. Used by **START** and **STOP**.
  - *Duration*: Estimated duration of the recording in frames.

> **Info:** For the setup of recording ports, please refer to section *Working with Mosart Port Configurations*. See also *Recording* in section *Viz Mosart User Interface* of the *Viz Mosart User Guide*.

- START: Viz Mosart will start a recording previously PREPAREd
  - *Parameter*: ,*<Recorder>*
  - *Recorder*: Name of recording session as created by PREPARE

> **Info:** Note the leading comma in ,*<Recorder>* as explained above.

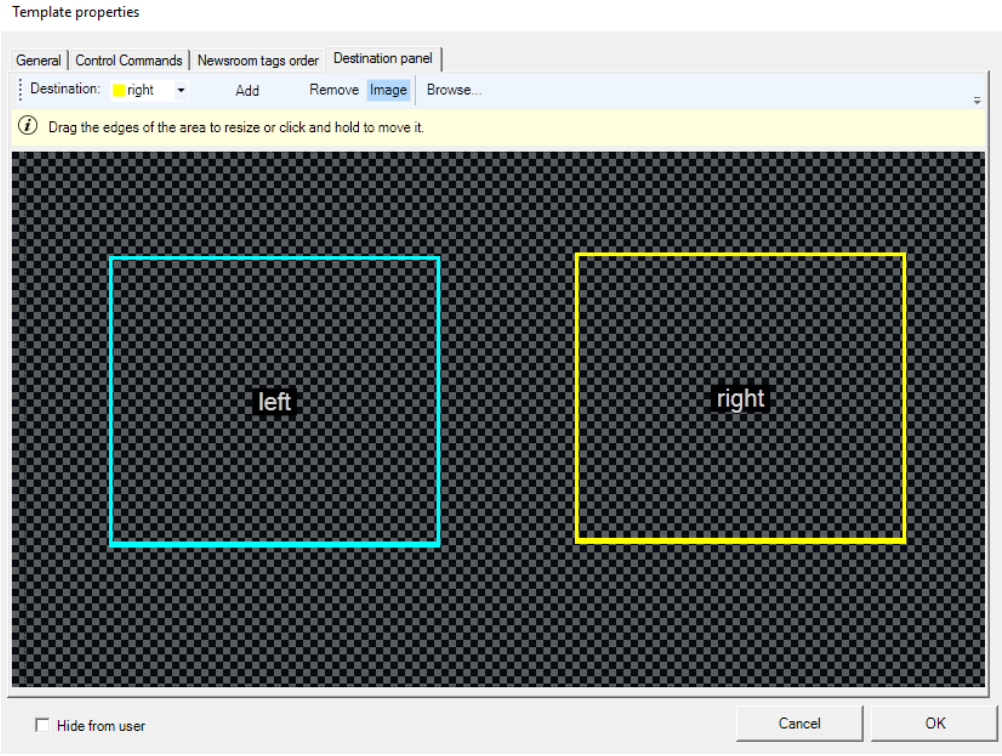
- STOP: Viz Mosart will stop a recording previously STARTed.
  - *Parameter*: ,*<Recorder>*
  - *Recorder*: Name of recording session as created by PREPARE

> **Info:** Note the leading comma in ,*<Recorder>* as explained above.

9.3.6 Destination Panel

In the **Destination panel** tab, you can create rectangles that are then available in the Viz Mosart UI for assigning sources to newsroom tag destinations, while the template is in either Preview or Program.

- Press **Add** to create a selected target area.
  The target areas display in the Viz Mosart UI as drop down boxes. They become clickable if a source or shortcut is selected by the user.



See also section *Program Window* in the *Viz Mosart User Guide*.

9.3.7 Empty Templates

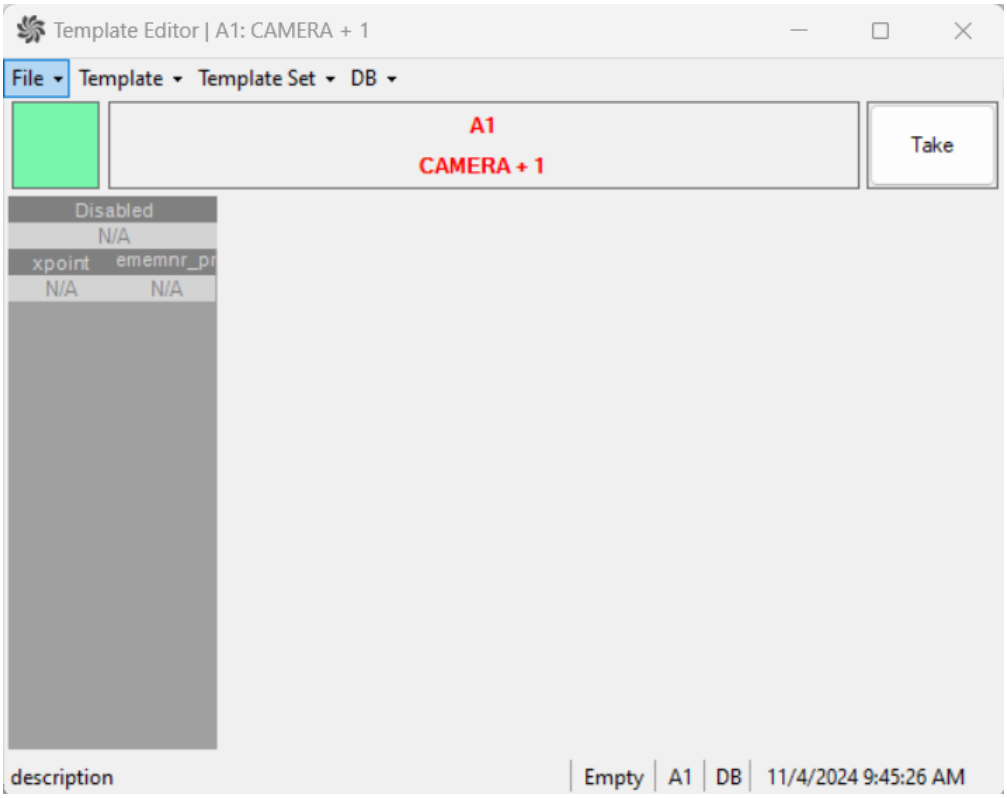
A hierarchical setup and the behavior of empty templates enables flexible template management across galleries, providing control over when templates are used directly or when Mosart should search for a non-empty version in the parent set.

Definition

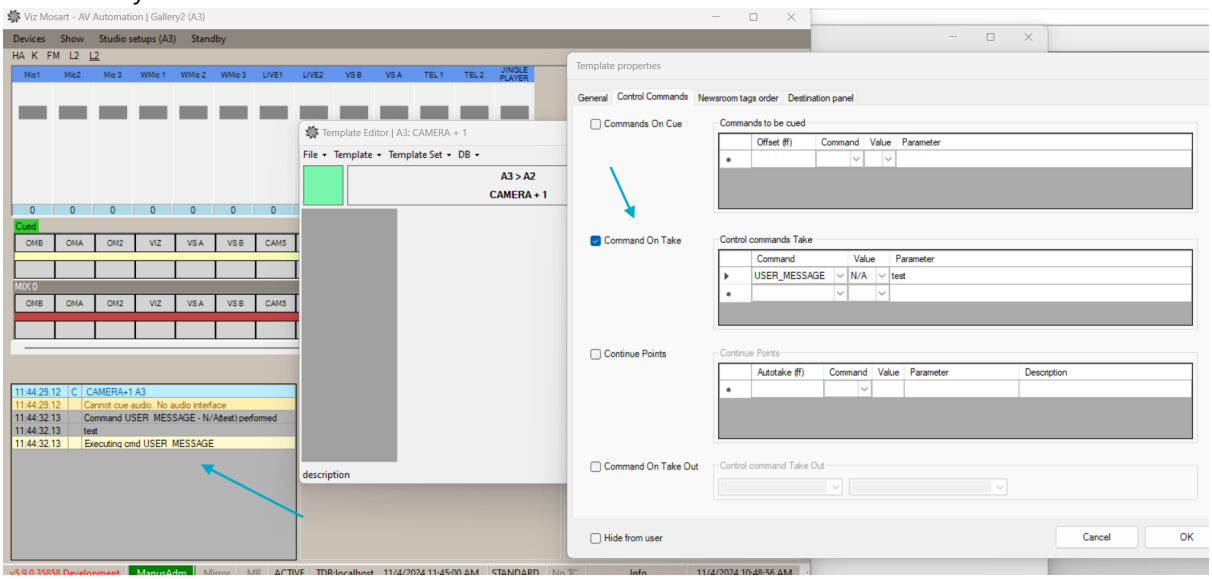
In Viz Mosart, an *empty template* is a template containing *no* actionable or configured elements for driving a broadcast production. A non-empty template is considered to be *implemented*.
Whether a template is considered empty or not, depends on the following characteristics:

- **No devices configured**: The template does not include any operational devices (for example, camera, audio, graphics) that would activate during a production.
  Since an empty template has no actionable functions, it cannot have newsroom tags or destination panel settings configured.

A template with the following content, where the *Switch crosspoint* device and the transition are disabled, is also considered empty, as it performs no switching functions.



- **No control commands**: If a template contains any control commands (for example, *commands on cue*, *on take*, or *on take out*), it is treated as *non-empty*, as these commands imply operational intent and functionality.



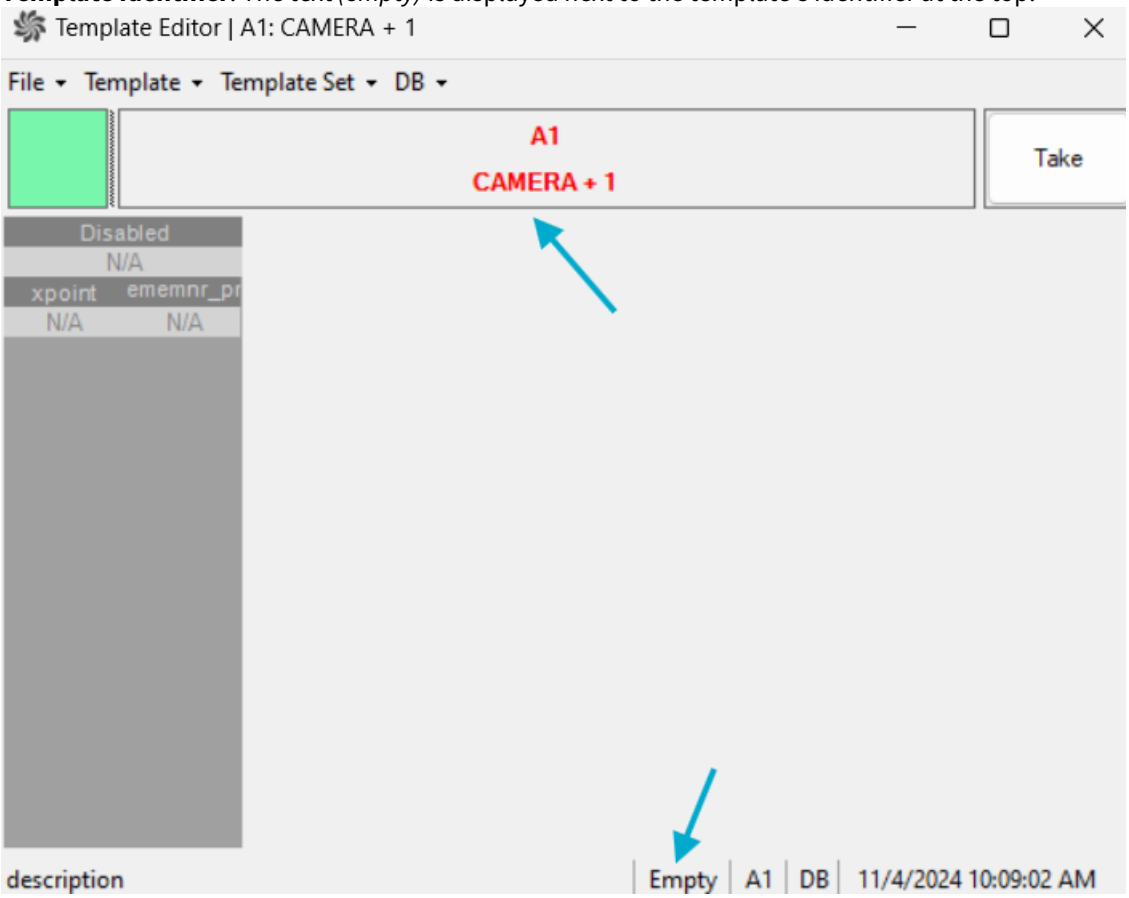
- **General properties**: General properties can still be set on an empty template. These settings don't change the empty status, as they involve *no* device or control functionality.
- **Break Templates**: A special case applies to a Break template. A template designated as a *break template* is automatically treated as *non-empty*, regardless of its content.

This is because break templates are inherently intended for specific actions within the broadcast and are therefore not considered empty.

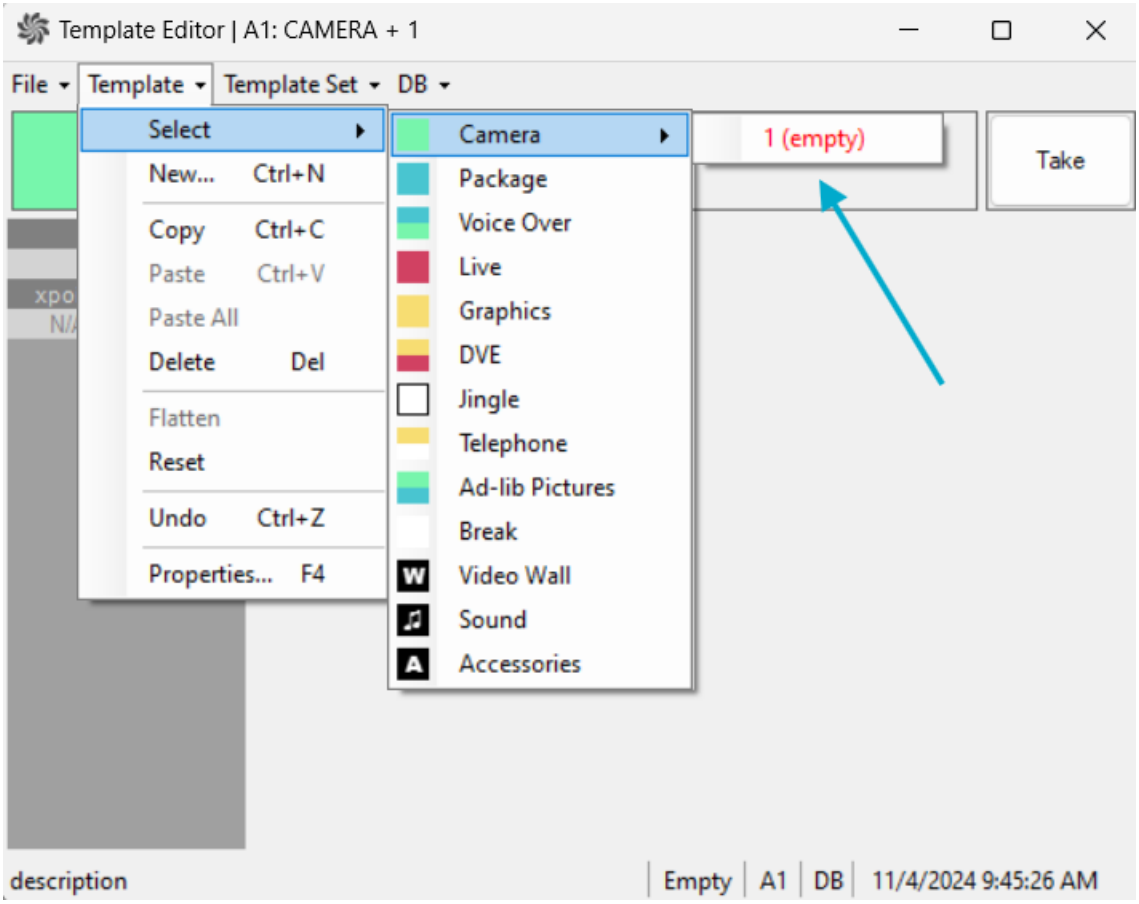
Visualization

In the Template Editor, there are several visual signs indicating an empty template:

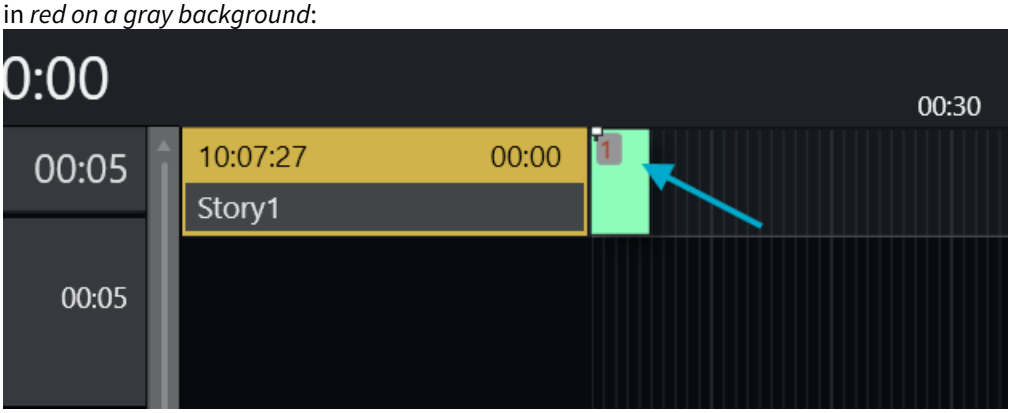
1. **Status Bar**: The label *Empty* appears in the status bar at the bottom of the editor.
2. **Template Identifier**: The text *(empty)* is displayed next to the template's identifier at the top.



3. **Template Selection Menu**: When searching for the template in the **Select** menu (as shown in the screenshot below), '*(empty)*' also appears beside the template name, immediately clarifying that the template is empty.



In the Mosart UI, a story item that refers to an empty template is indicated by showing the template variant in *red on a gray background*:



These visual indicators help users quickly identify and confirm an empty template within the editor.

Working with Empty Templates in Viz Mosart

When a template is created in one gallery within a shared template set, an empty version of that template is automatically created in the shared template set across all other galleries.
This approach encourages each gallery to individually implement or customize these templates as required. Tools

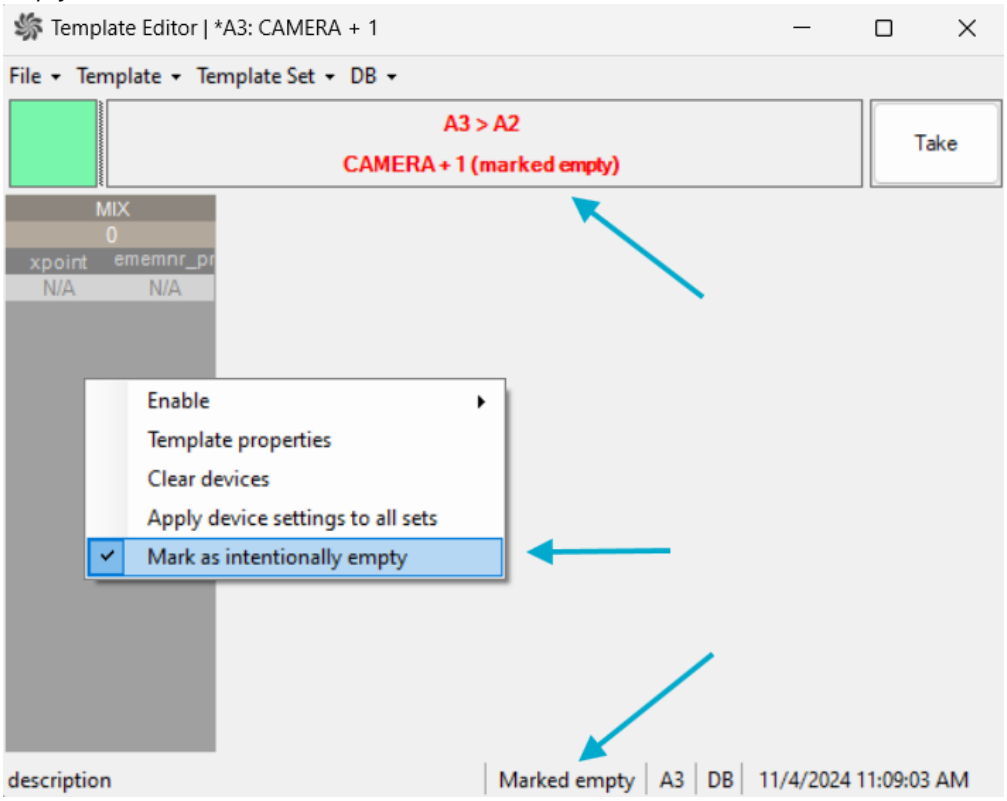
such as copying the implementation from another gallery (found in the **DB** menu of the **Template Editor**) can simplify this process by allowing users to replicate template configurations between galleries.

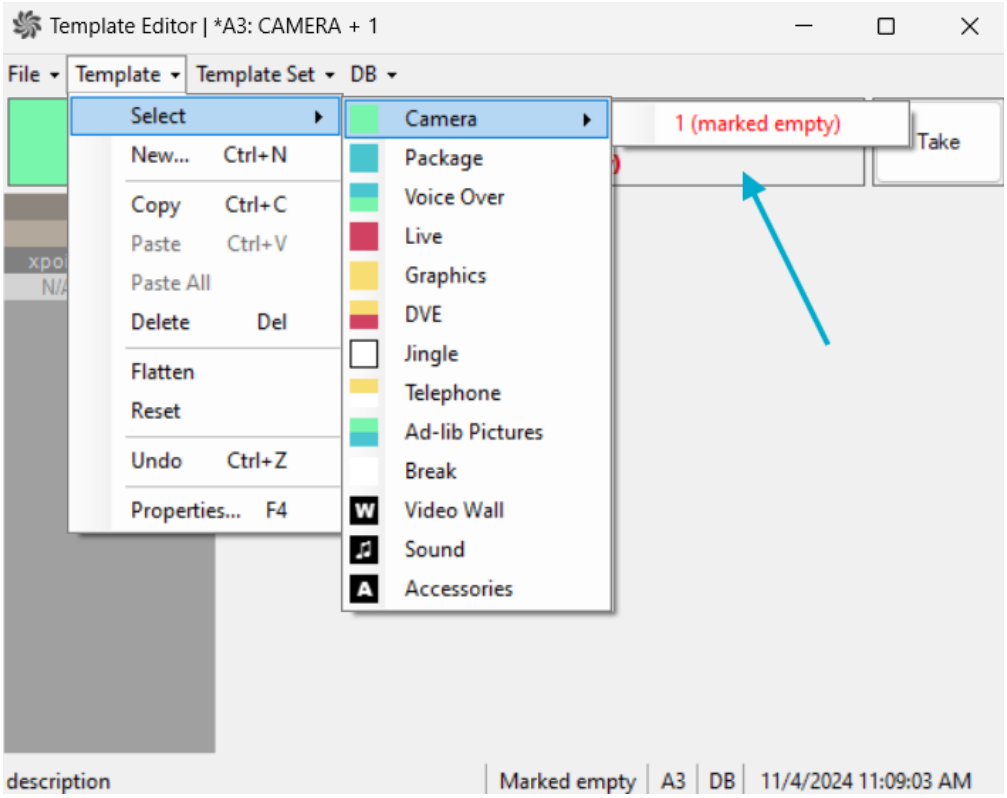
Hierarchical Template Setup

'Mark Template as Empty' Feature

In a hierarchical template setup, the behavior of empty templates depends on whether the setting **Use 'Mark template as empty' feature** is enabled in the Manus Admin settings.

1. When **Use 'Mark template as empty' feature** is enabled:
   - An option is made available on an empty template in *Template Editor* to mark it as *intentionally empty*:





- A template intentionally *marked as empty*, is processed with this quality during execution and visualization.
  Any inherited implementations of the template from a parent set are ignored, effectively canceling the function of the base template.
- When a template is *not* marked as intentionally empty, Mosart searches upwards in the template set hierarchy to find the next available non-empty template, which is then used for execution and visualization.

> ⚠ The option **Mark as Intentionally Empty** is available only for an *empty* template. If a template marked as intentionally empty later has control commands and/or device functions added to it, the template will then be considered non-empty and the **Mark as Intentionally Empty** setting is ignored. The newly added control commands and/or device functions are executed in a standard way.

2. When **Use 'Mark template as empty' feature** is disabled:
   - Mosart processes any empty template as is, without searching for a non-empty version in the parent template set.
     The empty template is used directly in execution, effectively canceling any inherited base template.

Examples

Consider the following database setup where sets S1, S2, and S3 are *shared across galleries*:

Gallery G1

- **Set S1**: Template T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)

- **Set S3** (based on S2): T1 (inherited)

Gallery G2

- **Set S1**: T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)
- **Set S3** (based on S2): T1 (inherited)

Gallery G3

- **Set S1**: T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)
- **Set S3** (based on S2): T1 (inherited)

If, in **Gallery G1, Set S3**, Template T1 is *overridden* by adding, for example an audio fader, this change creates an empty override of T1 in all other galleries. This approach assumes that each gallery will implement the template individually as needed.

After this action, the database looks like following:

Gallery G1

- **Set S1**: Template T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)
- **Set S3** (based on S2): T1 (**override**)

Gallery G2

- **Set S1**: T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)
- **Set S3** (based on S2): T1 (**empty**)

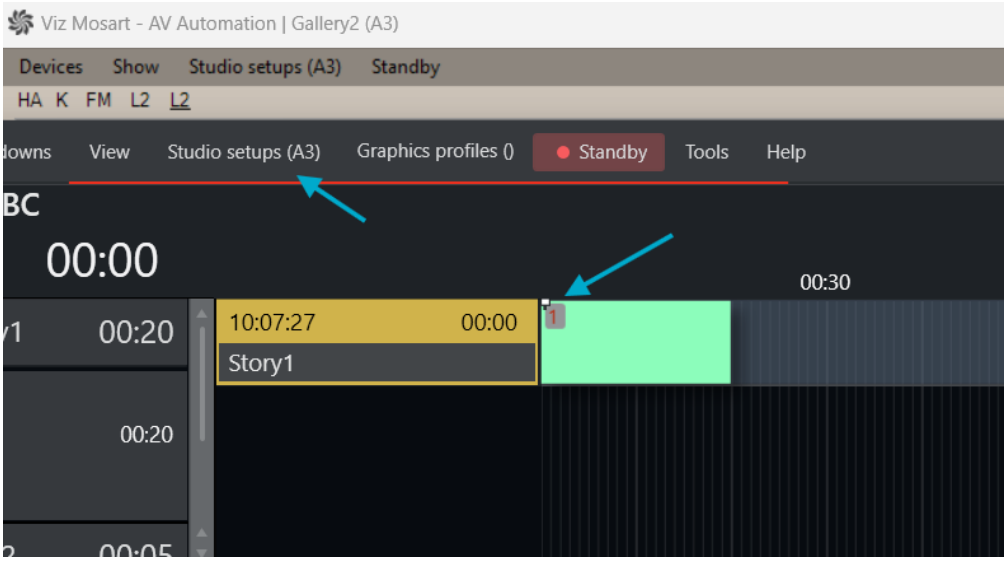
Gallery G3

- **Set S1**: T1 (not empty)
- **Set S2** (based on S1): T1 (inherited)
- **Set S3** (based on S2): T1 (**empty**)

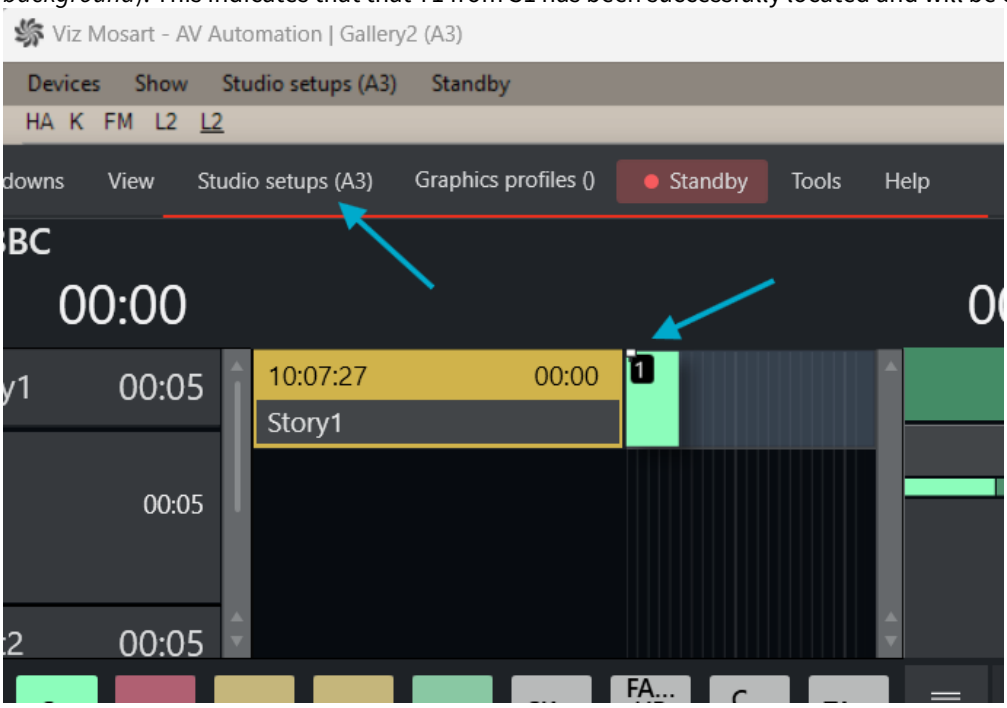
This could lead to issues in **Galleries G2 and G3** if the template remains empty and unimplemented. Since an empty template provides no executable actions, Mosart would not perform any actions associated with that template if used in a rundown.

Execution behavior of *Mark Template as Empty* feature

1. When ***Use 'Mark template as empty' feature*** is *disabled*:
   In **Gallery G2 and G3, Set S3**, Template T1 will remain empty in Mosart's execution and display. The template appears empty in the Mosart UI, with no alternative fallback template used for execution.



2. When ***Use 'Mark template as empty' feature*** is *enabled*:
   Mosart will search the parent template chain to locate the next non-empty version of T1.
   For **Gallery G3, Set S3**, Mosart will find and use T1 from **Set S1** (non-empty) for execution and display.
   In the Mosart UI, this will display as a *found* template, indicated by a *white variant number on a black background*). This indicates that that T1 from S1 has been successfully located and will be executed.



3. When ***Use 'Mark template as empty' feature*** is *enabled* and the template is marked as *intentionally empty*.
   The template will behave the same as in scenario 1 above, remaining empty in Mosart's execution and display.

9.3.8 Configuring a REST-based Driver

- Template Commands
- Placeholders
- Command Management
- Troubleshooting

Working with a REST-based Driver

Viz Mosart offers an easy to create, generic driver which can send REST commands to an external device. The initial driver construction is explained in section Generic REST Driver.

This section explains how to communicate with a REST-based driver from a template.

Tip:
- A two minute introduction video for using the REST API is **here.**
- For sample implementations, see section Generic REST Drivers.

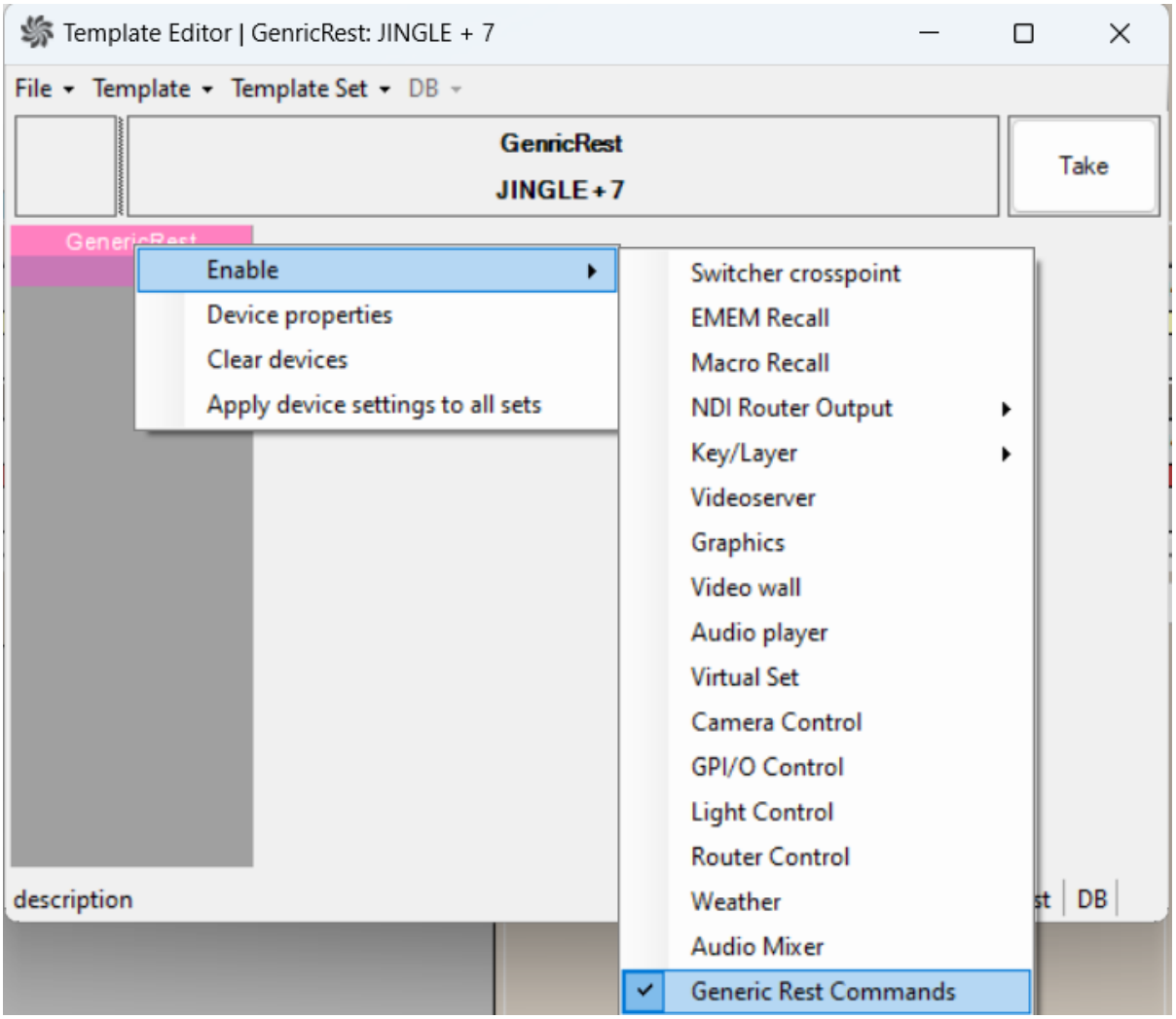
Template Commands

To configure REST commands

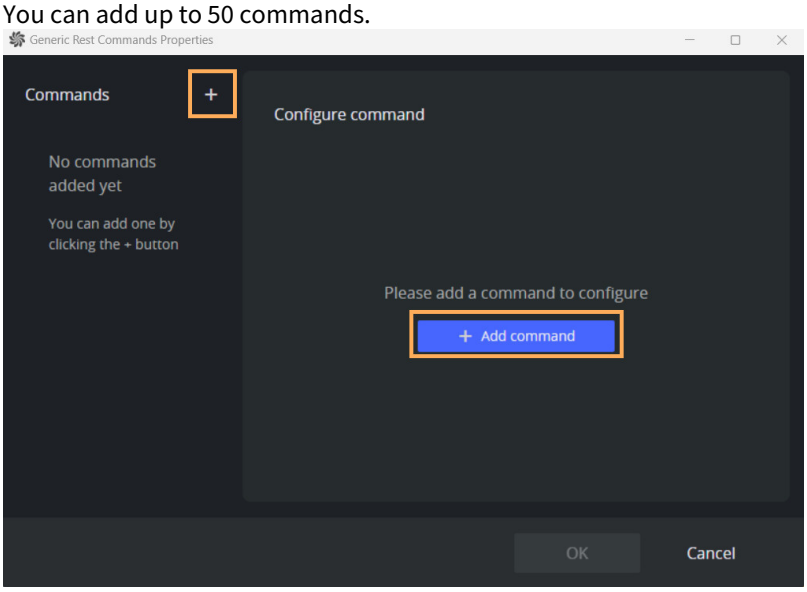
Note: Before following the procedure below, you must first have some Generic REST device configured. This is explained in section Generic REST Drivers.

Begin by globally enabling Generic Rest Commands:

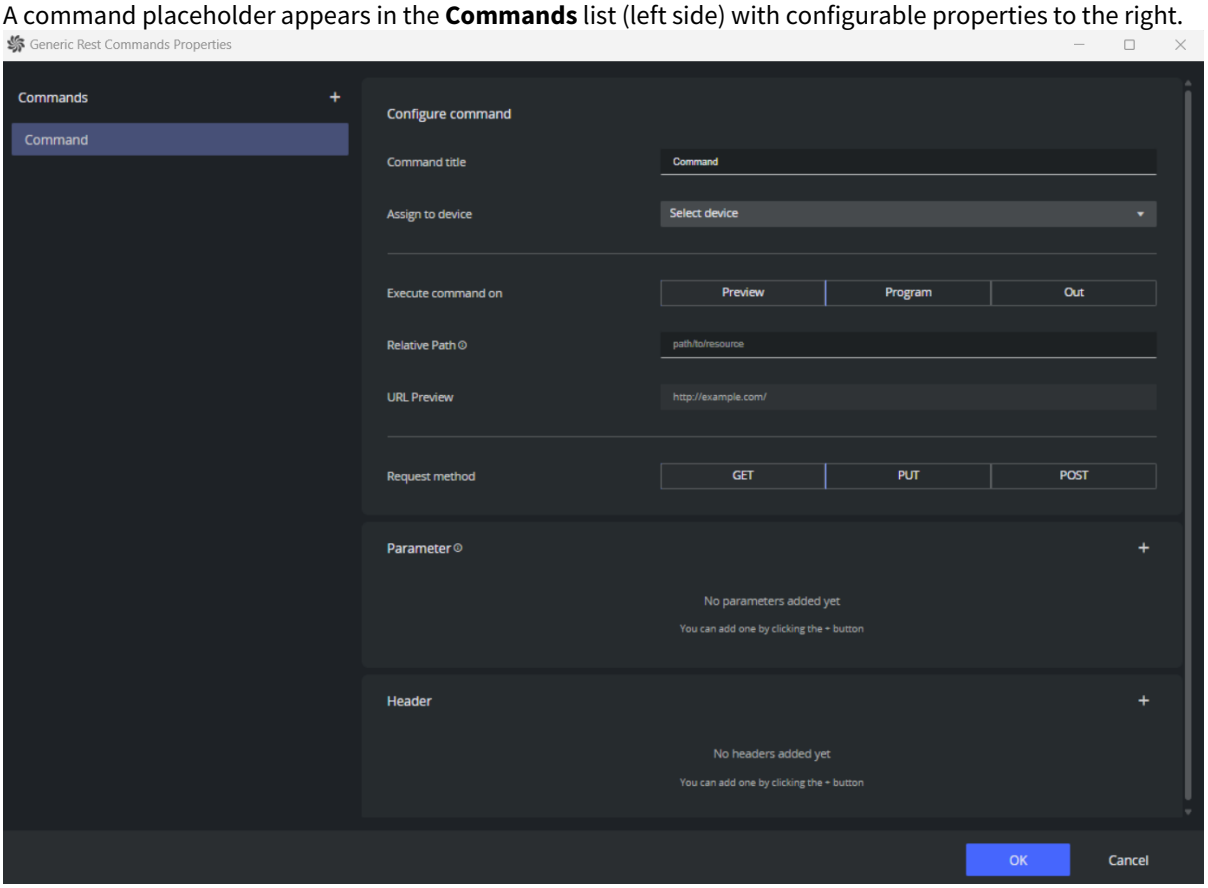
1. Navigate to **AV Automation menu > Devices > Template editor**.
2. Select or create a new template.
3. In the device's Function area (grey/black to the left), right-click and select **Enable > Generic REST Commands**.



4. In the **Generic REST Commands Properties** menu that appears, click the Plus (**+**) sign or **Add command** to add a new REST command.
   You can add up to 50 commands.

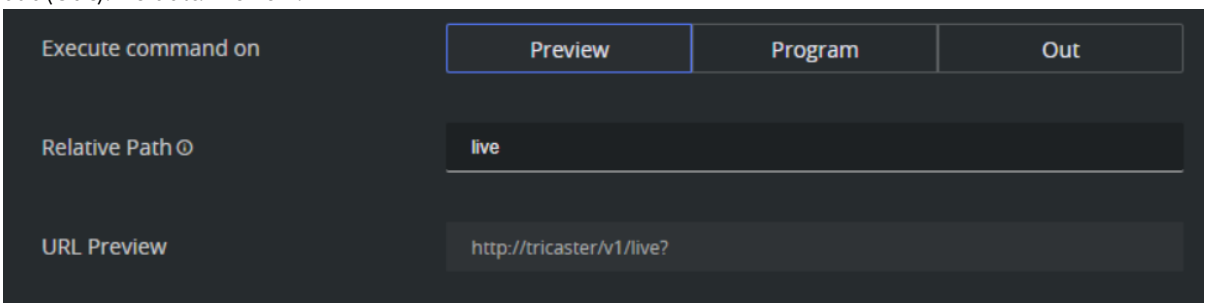


5. A command placeholder appears in the **Commands** list (left side) with configurable properties to the right.



6. Configure the properties for a Generic REST command as follow:

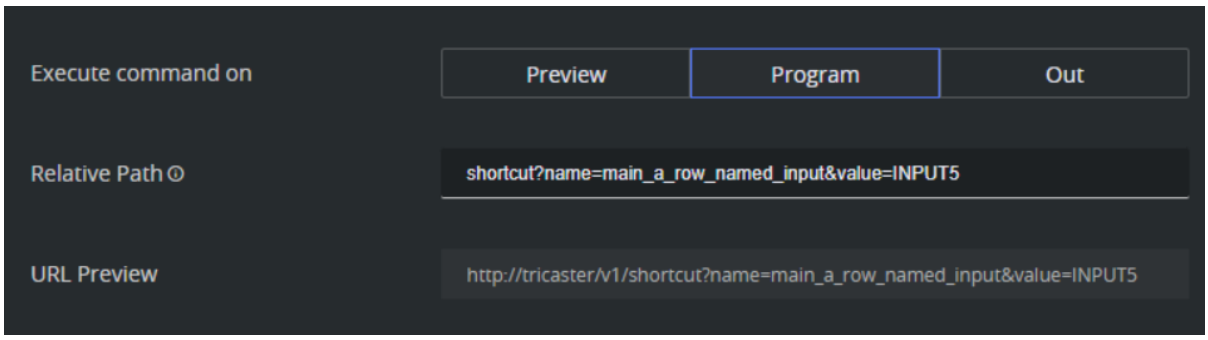
- **Command title**: The name that identifies the generic REST command. Default: *Command*.
- **Assign to device**: Select the Generic REST device configured in Mosart Administrator Settings to execute the command.
    - If no devices are available, you must first define your REST driver.
    - If no device is selected, the command will not be executed. Default: *empty string*.
- **Execute command on**: Specifies when Viz Mosart should execute the command, relative to the template lifecycle.
  Options include when the template is *cued* (**Preview**), *taken* on-air (**Program**) or when the template is *taken out* (**Out**). Default: *Preview*.



- **Relative Path**: The resource path appended to the *Base URL* for the selected device.

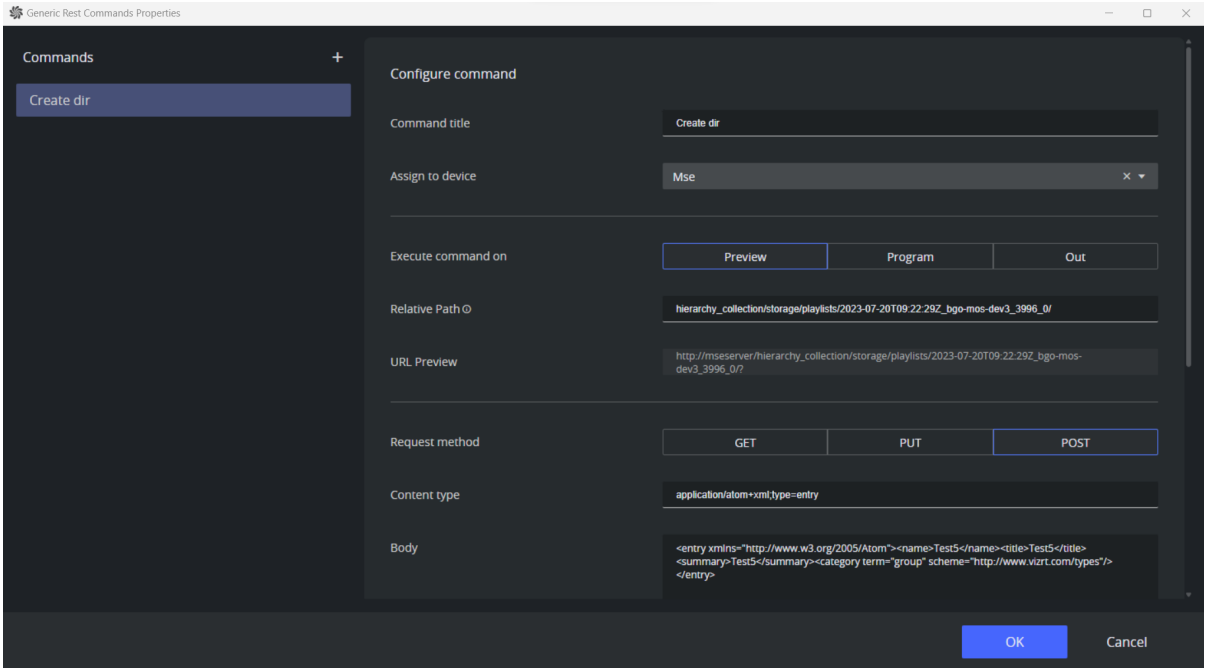
Note: This path should *not* start with a scheme (for example, http://) or a leading slash.

- The path can optionally include query string parameters. For example, if a parameter value does not require a newsroom tag, it can be added directly to the value of Relative Path.
- A URL parameter contains a *key* and a *value*, separated by an equal sign (=).
  A URL can contain multiple parameters, which are then each separated with an ampersand (*&*).

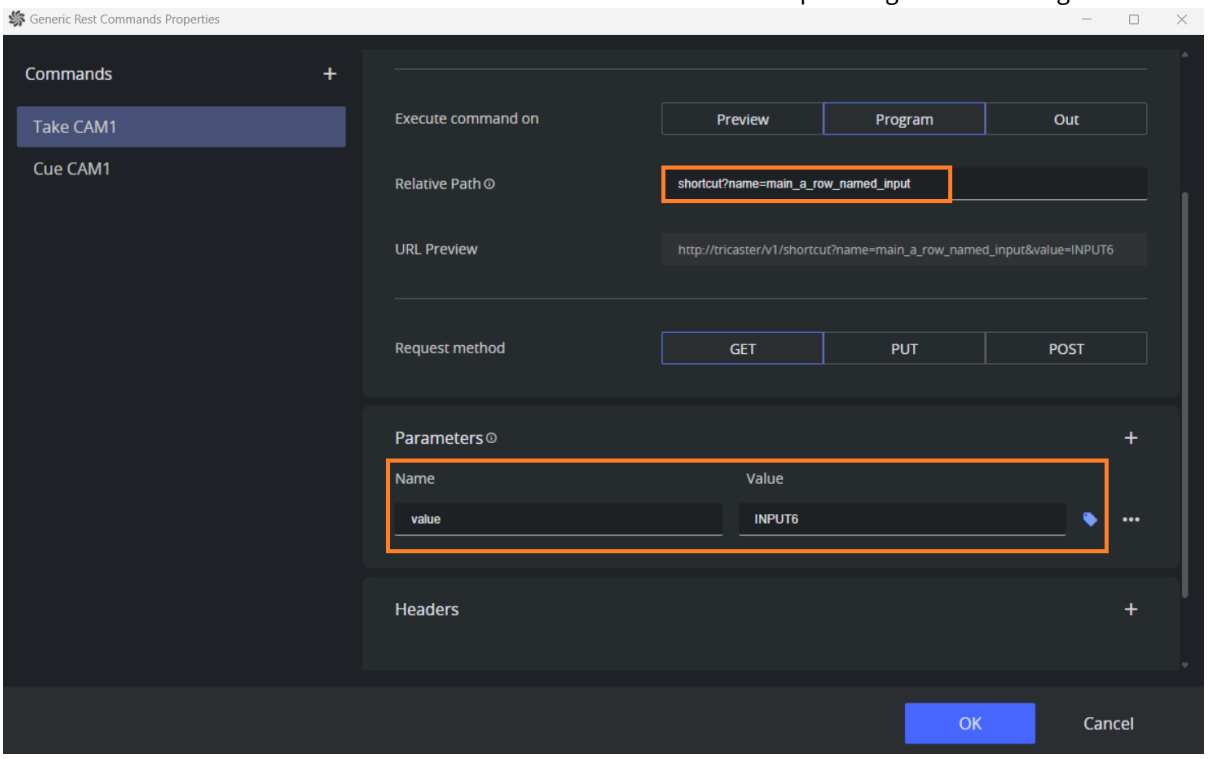


- **URL Preview:** This field displays a preview of the complete URL, composed of the *Base URL*, **Relative Path**, and any extra parameters specified in the **Parameters** section.
- **Request method**: The HTTP method used for the request.
  Options are: *GET*, *PUT*, and *POST*. For PUT or POST methods, the *Content Type* and *Body* of the request can be configured.
- **Content type**: Specifies the value for the *Content-Type* HTTP header, indicating media type of the resource or data in the request body.
  Required for *PUT* and *POST* methods.
  Common Content-Type values include *text/plain*, *text/xml*, *application/xml*, *application/json*.
  The *charset* parameter can be used to set the character encoding for the text being sent. Default: *text/plain; charset=utf-8*.
- **Body**: The data sent in the request body. Required for *PUT* and *POST* methods. *Body* must be formatted according to the specified *Content Type*.

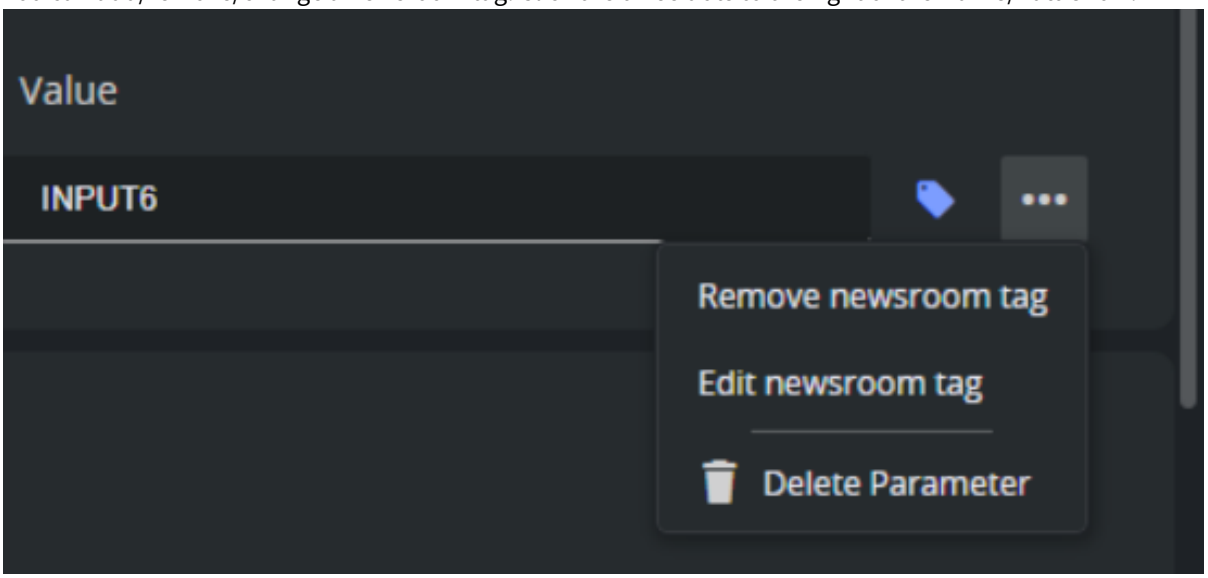
Copyright © 2026 Vizrt



- **Parameter**: Additional parameters can be included in the command request by adding them in this list using the Plus (**+**) sign. This gives the possibility of providing *newsroom tags* for the parameters' values, enabling dynamic control from the newsroom control system (NRCS).
  In the following example, two parameters are added to the request command.
  - Parameter **Name** is attached to the *Relative Path* with a static value *main_a_named_input*.
  - Parameter **Value** is added in the *Parameters* section with static value *INPUT6* and a newsroom tag attached. This enables the value to be overwritten from the NRCS when planning the show design.



Newsroom tags
You can add/remove/change a newsroom tag. Click the three dots to the right of the **Name**/**Value** row:



If a parameter is included in both **Relative Path** and the **Parameters** listing, the value in **Parameters** will take precedence, overwriting the value in **Relative Path**.

- **Header**: To set authentication tokens, cache control, or other custom headers required by the REST server, additional HTTP headers can be included in the request.
  Headers get added in a similar way as *Parameters*, with an option of using a newsroom tag as a header value.

Placeholders

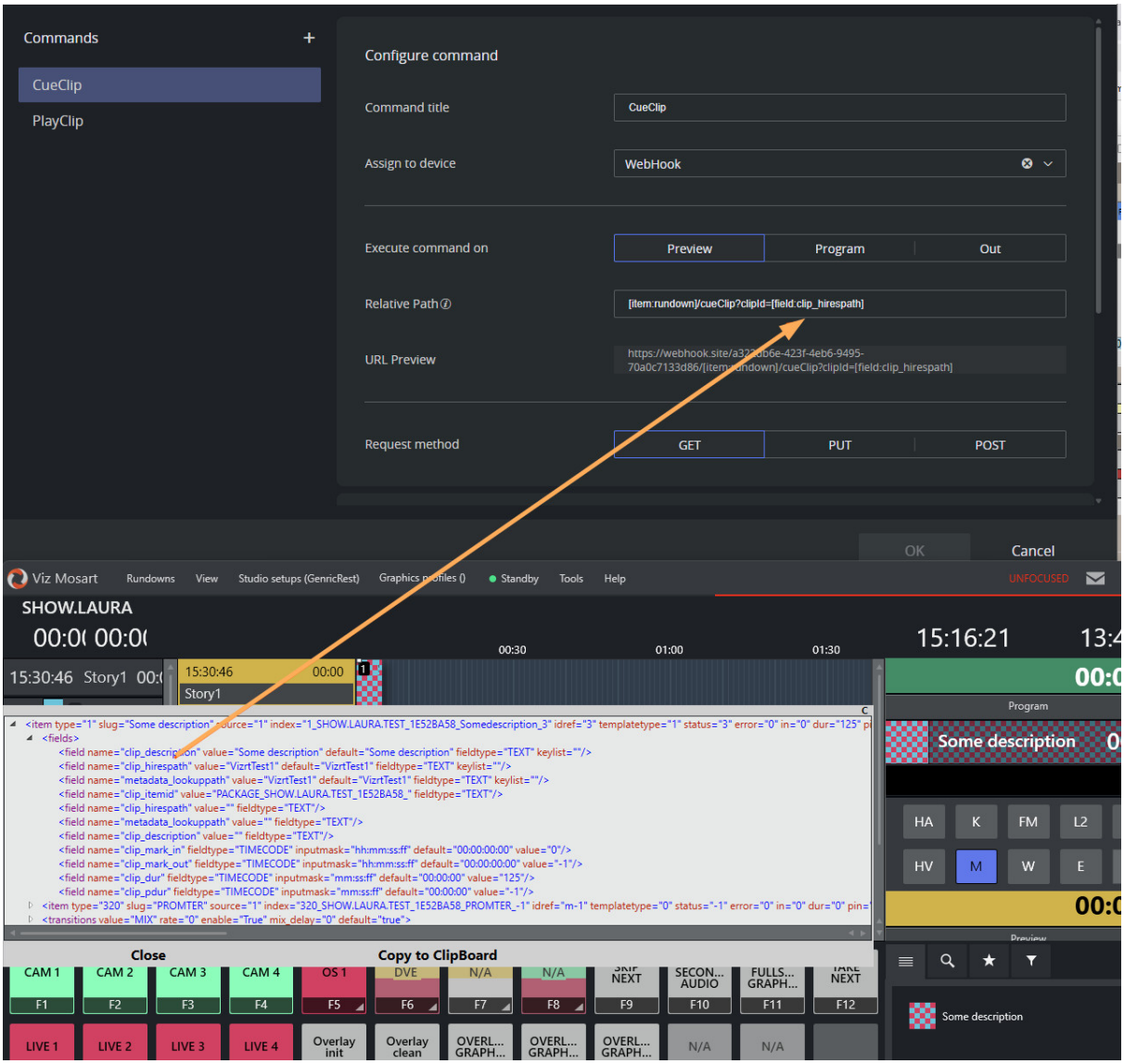
Placeholders let you insert dynamic data into the REST URL, headers or body. During show execution, these placeholders are replaced with current values just before the REST command is sent.
A placeholder can hold, for example:

- Data from an NRCS, passed as attributes and fields to a Mosart item.
- An XPath extraction relative to a Mosart item.
- The Current/Next story information.

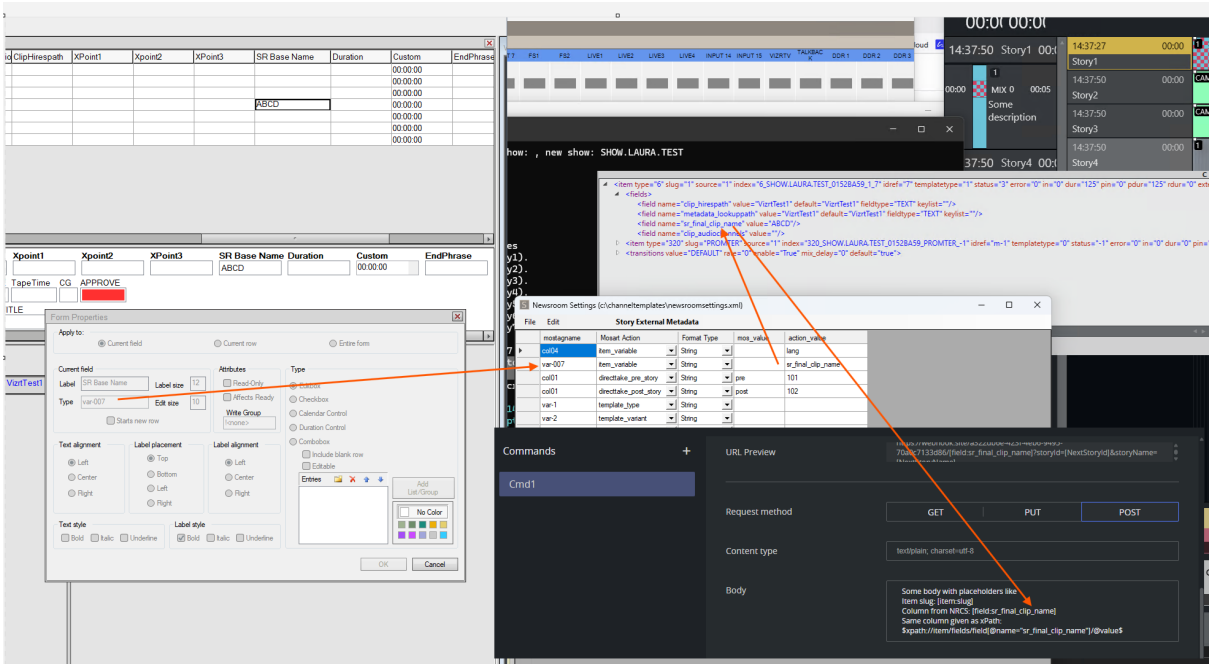
Supported placeholders:

- **[field:fieldName]**
  This placeholder represents data from the NRCS that is saved in the Mosart rundown as a field under a story item. Data from the NRCS can be passed to a Mosart story item in several ways:
  - An NRCS column mapped in Mosart Administrator's Story External Metadata as an `item_variable`.
  - A template variable, known as a *newsroom tag*, exposed to the NRCS using either the Mosart NRCS plugin or the MOS template object sent to NRCS.
  - Alternatively, Field Mapping can be used to extract values from a MOS object received from the NRCS and insert them into Mosart fields.

  In the example below, the placeholder `[field:clip_hirespath]` is used as the value for the `clipId` parameter in a URL. The field `clip_hirespath` is defined as a *newsroom tag* in the template, enabling the clip ID specified in the NRCS to be inserted into the corresponding Mosart item.



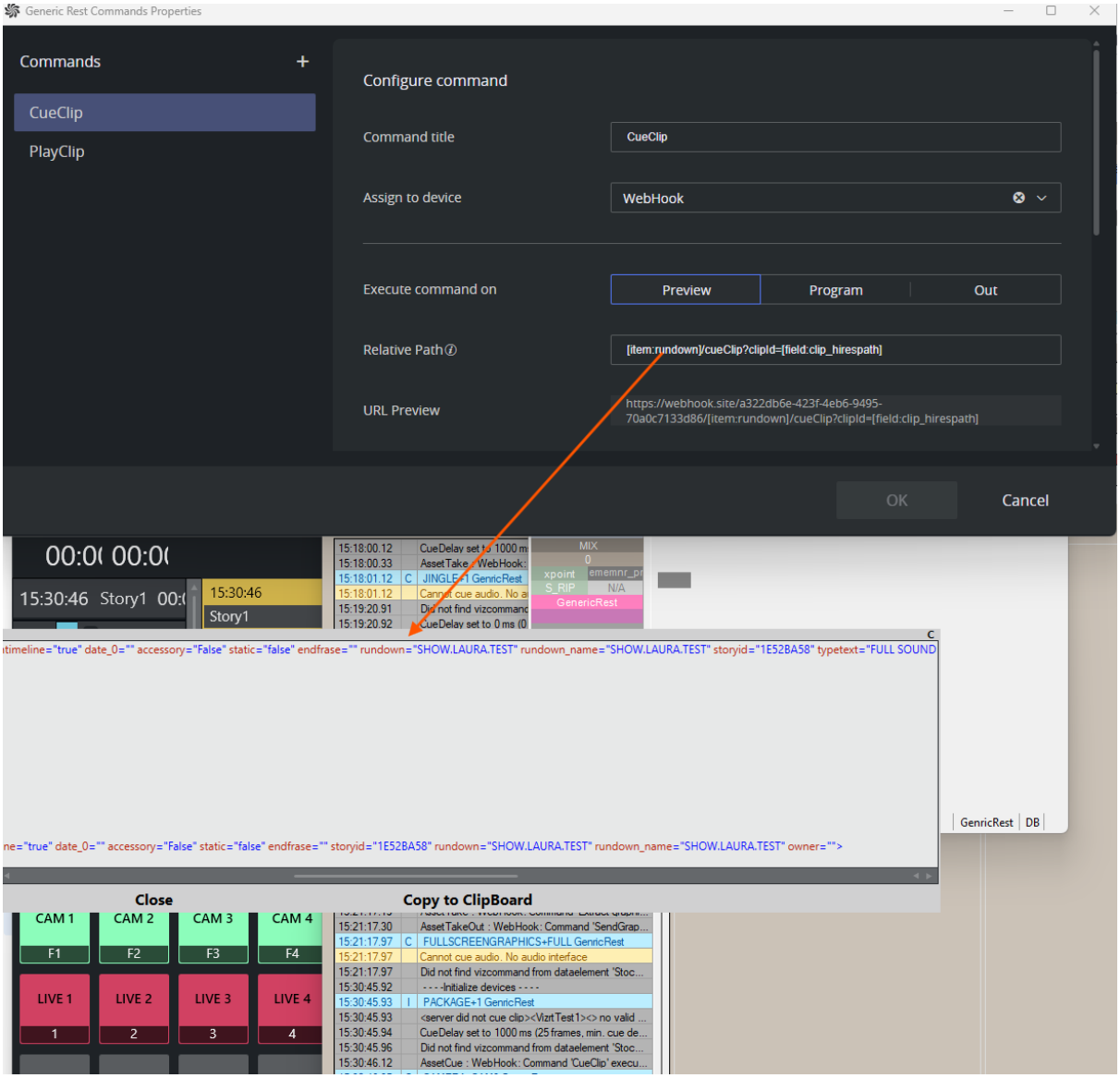
In the next example below, the field *sr_final_clip_name* inserts data from an NRCS column into the Mosart item:



- **[item:attributeName]**

  An attribute from the `<item>` XML representing a Mosart item. For example, `[item:type]`.

  In the screenshot below, the rundown ID found as an attribute in a Mosart item, is passed to the URL REST call as a sub-path:



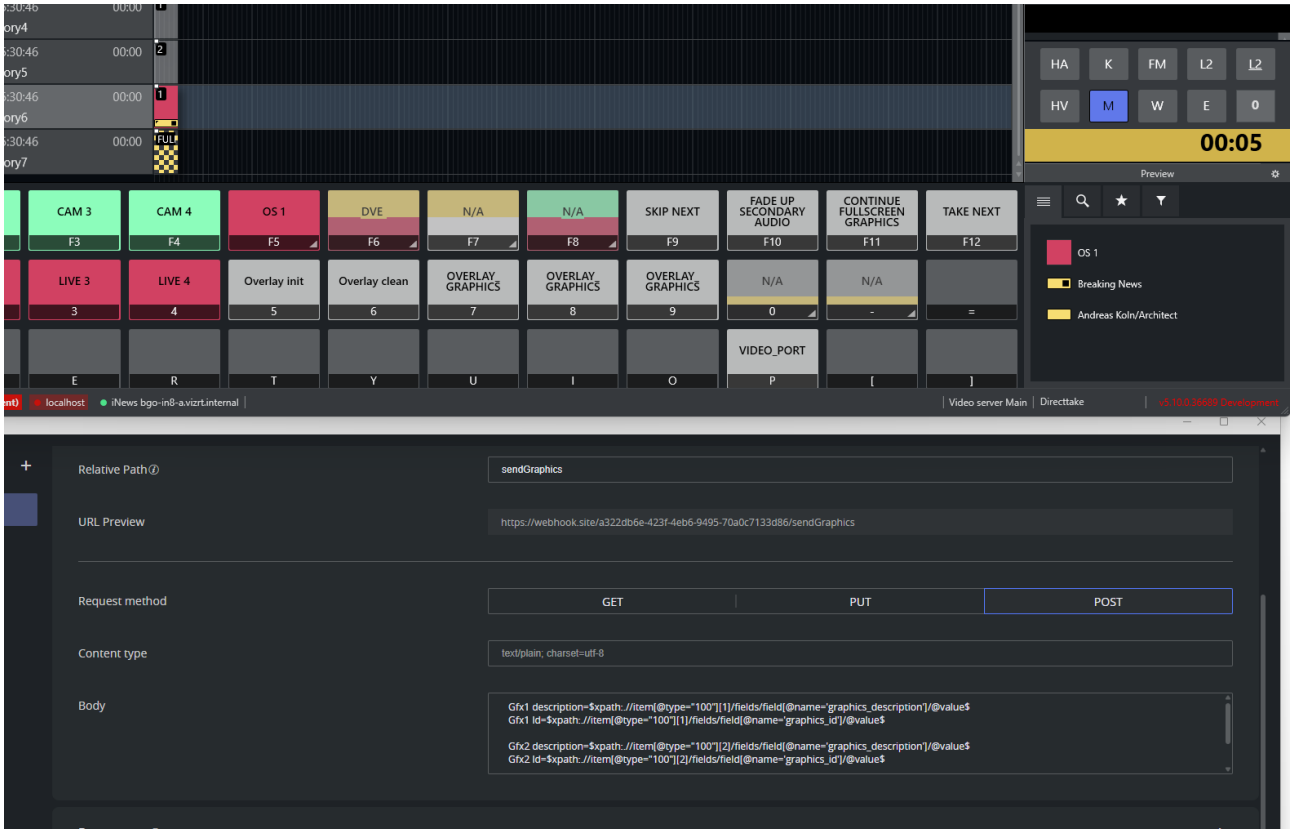
- **Predefined placeholders**
  - `[CurrentStoryId]`, `[CurrentStoryName]`: ID and name of the on-air story.
  - `[NextStoryId]`, `[NextStoryName]`: ID and name of the Mosart story marked as *Set as Next*.
- **$xpath:expression$**

  With an XPath query, you can extract data from the `<item>` XML which is representing a Mosart rundown item.

  Example: `$xpath:/item/fields/field[@name='graphics_id']/@value$`.

> **Note**: Use `$...$` as XPath placeholders because XPath expressions may contain square brackets. However brackets around the placeholder could cause parsing conflicts.

In the example below, we send information about the *overlay graphics* found on a story item:

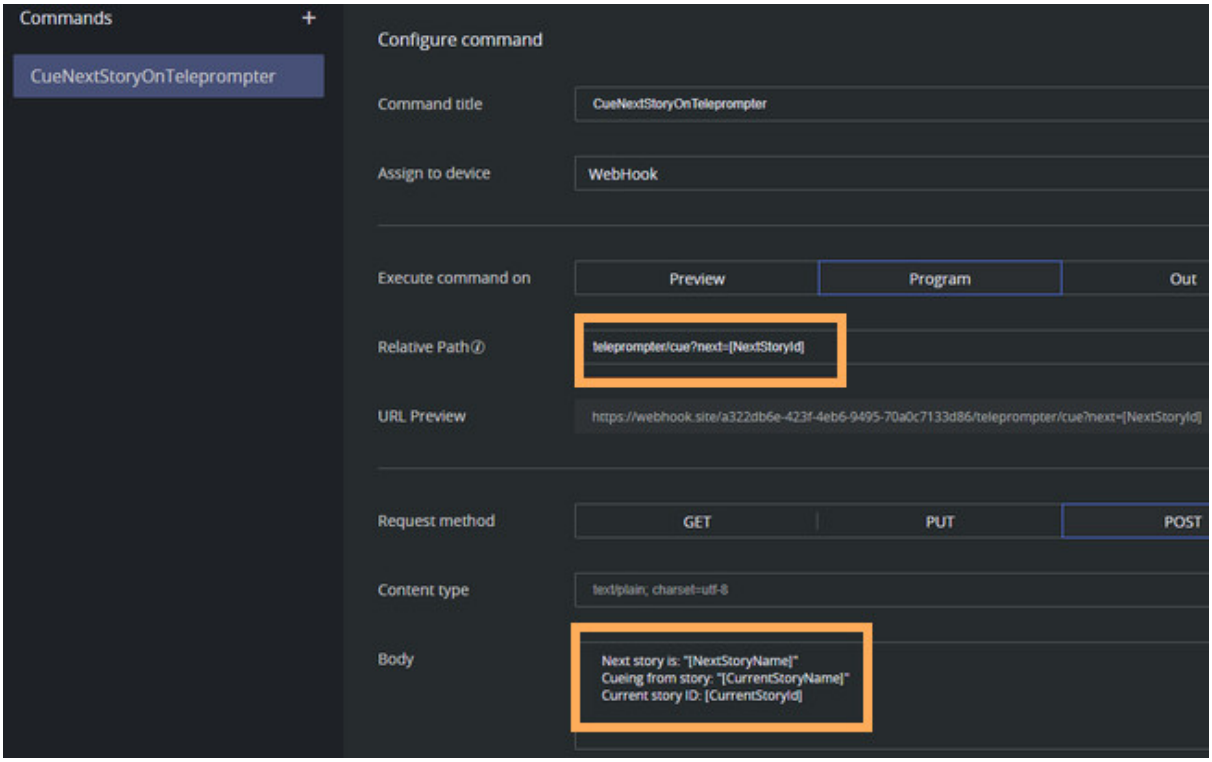


Note: The **Parameters** field does not support *placeholders* (for example, `[NextStoryId]` or `$xpath:expression$`) except for those originating from newsroom tags, which in fact has the same result as a `[field:fieldName] placeholder`. If you need to use other types of placeholders, place them directly in **Relative Path**. See examples below:

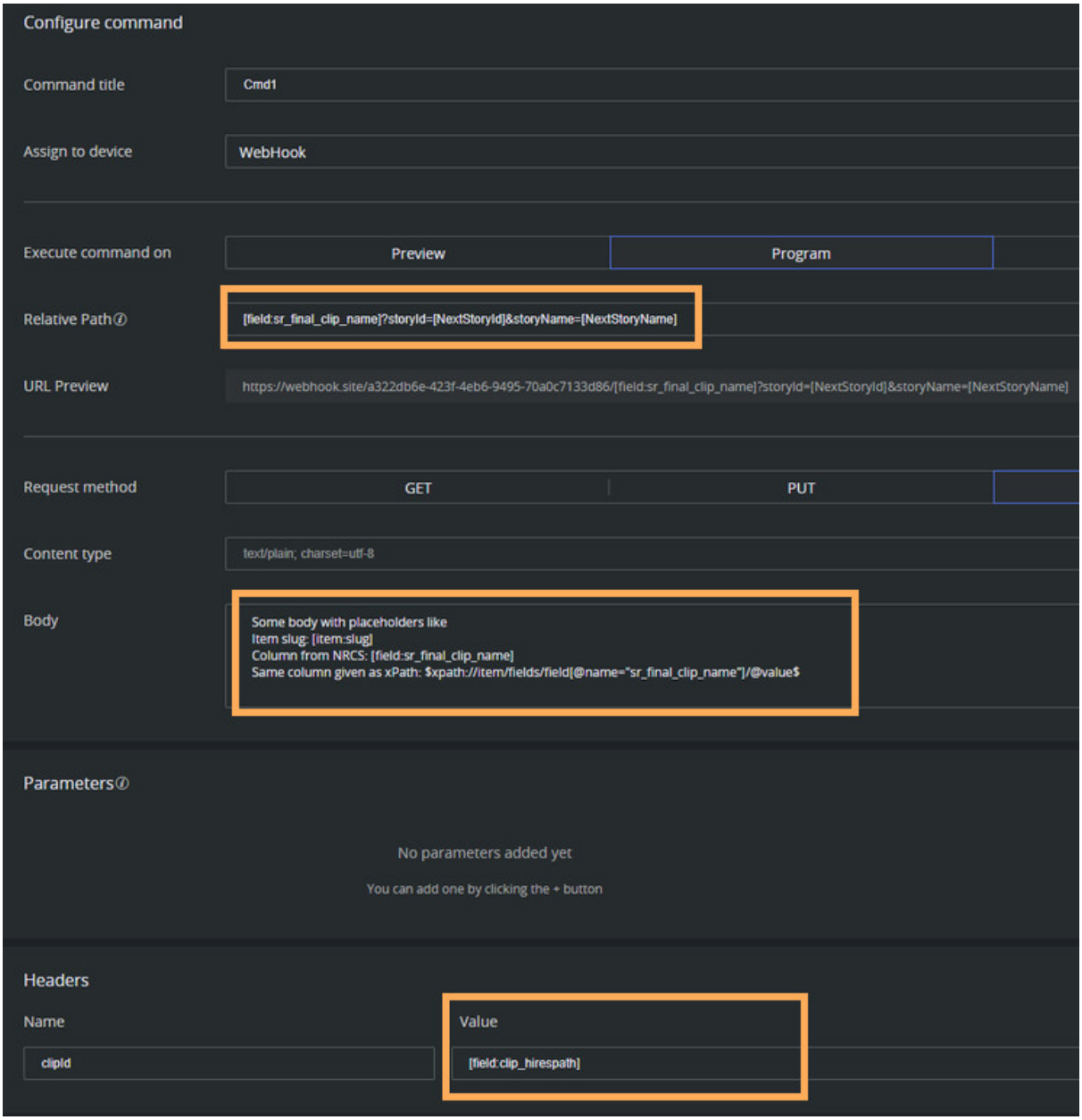
Examples

Sending the next story ID to a teleprompter

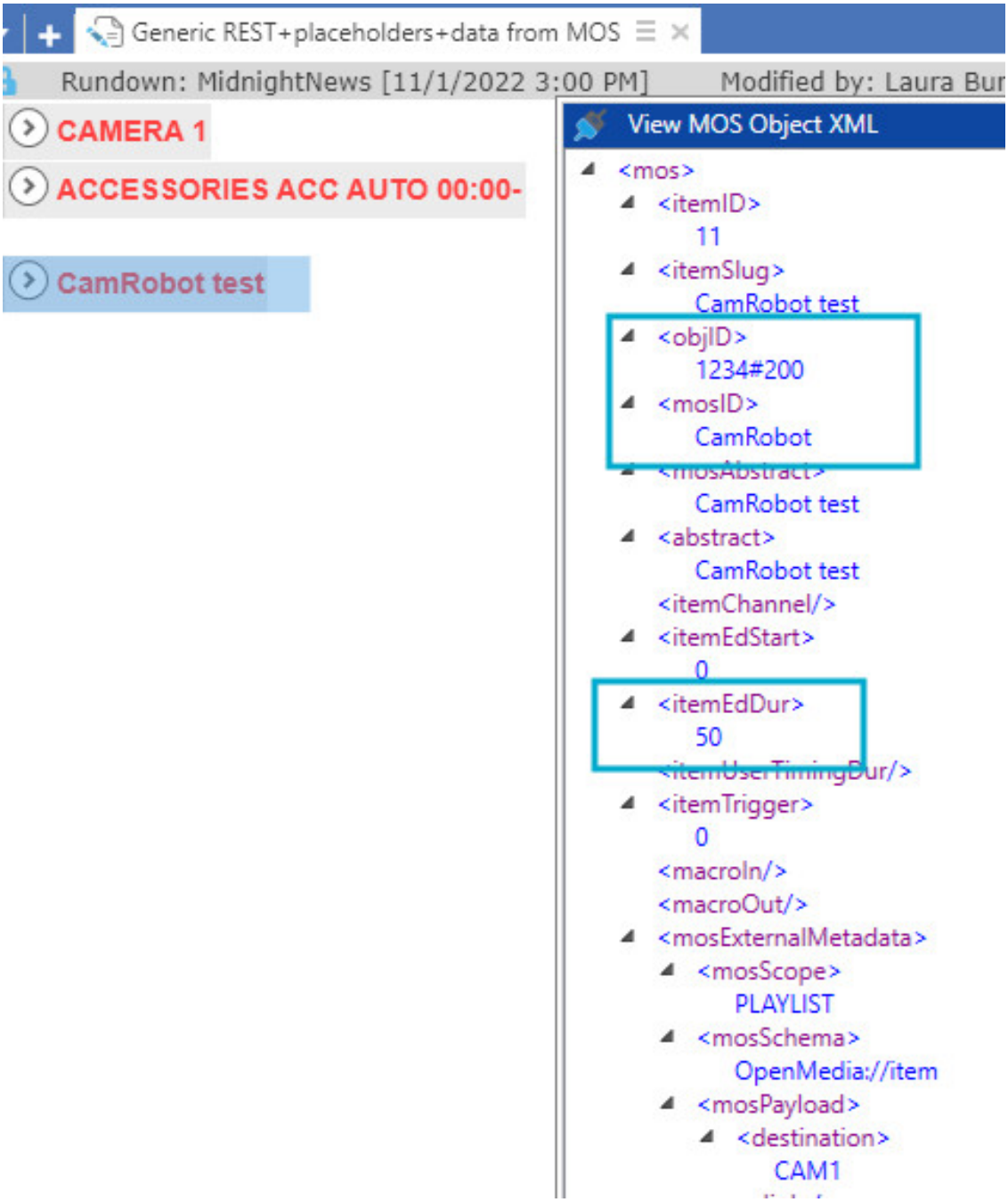
1. On a camera template, configure a Generic REST command on Program to include the next story ID using the predefined placeholder *NextStoryId,* either in the request path or in the body as supported by the teleprompter REST endpoint.



2. Placeholders defined as a *relative path*, *sub-path* or *parameter* in either the Body or Headers:



3. Pass the data from the NRCS through a MOS object into a Generic REST command configured on a Mosart template.
   As an example, instructions on movement and duration of a selected robotic camera is provided from the NRCS.
   The NRCS instructions detailing the camera ID and movement duration is provided through a *MOS object* that Viz Mosart does not by default, support.
   If the device has support for REST API, these instructions can be executed with generic REST commands, as illustrated below.
   a. In the NRCS, build a story containing a Mosart Primary template and an empty *Accessories* template, followed by the MOS object that contains the robotic camera details.
      The MOS object is unknown to Viz Mosart, so we attach the MOS object data in an *Accessories* template.

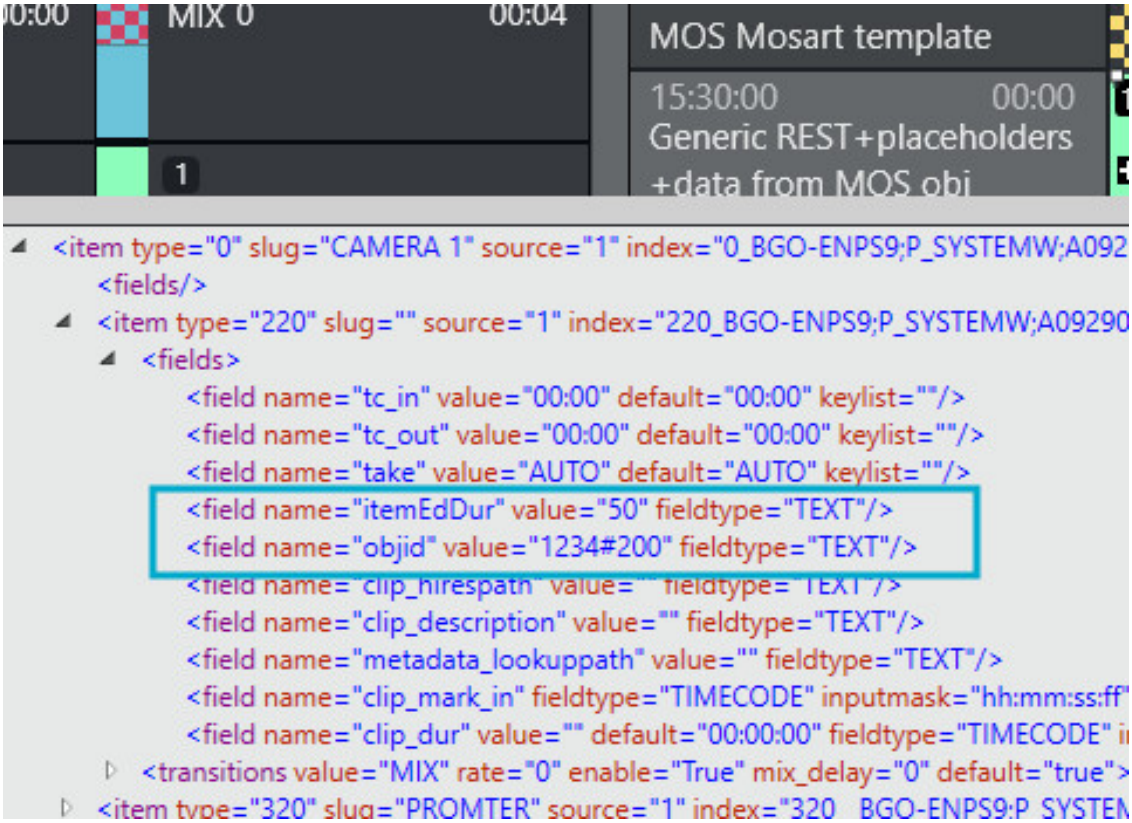


b. Use the *Field Mapping* in file newsroomsettings.xml (located on the Mosart server) to attach the MOS object data to the ACCESSORIES template.

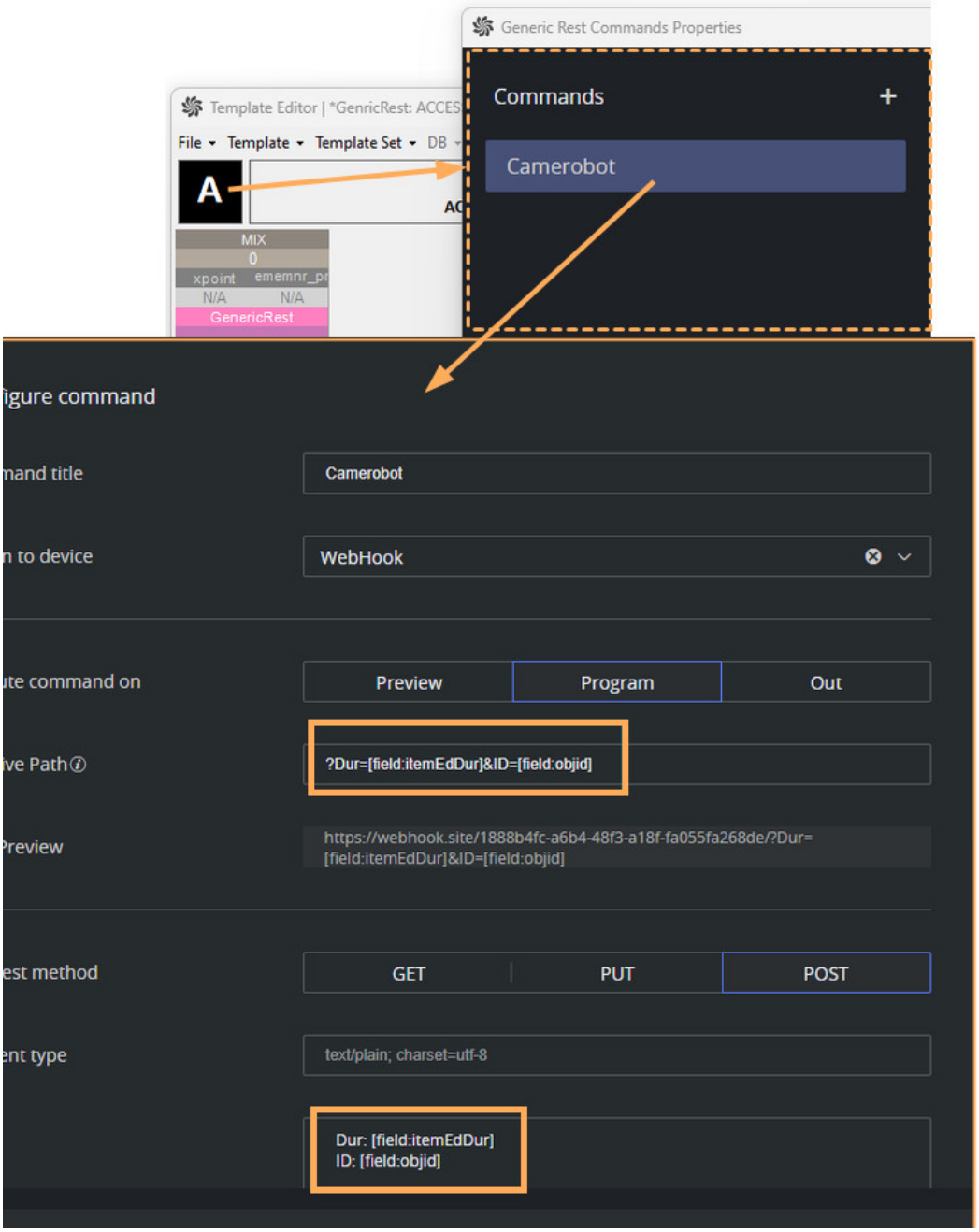
```
<mosids>
  <tag name="PILOT" value="PILOT" appendContent="true" fieldMapping="false" embeddedMo
  <tag name="vizrt.wall" value="VIZW" appendContent="true" fieldMapping="false" embedd
  <tag name="clipdalet.mos" value="DALET.CLIP" appendContent="true" embeddedMosartItem
  <tag name="mosart1.mos" value="DALET" appendContent="false" />
  <tag name="CamRobot" value="CamRobot" appendContent="false" fieldMapping="true" />
  <tag name="VizONE.mos" value="VIZONE" fieldMapping="true" appendContent="true" embed
  <tag name="VizONE2.mos" value="VIZONE2" fieldMapping="true" appendContent="true" emb
  <tag name="mosart.roger.mos" value="OCTOPUS.CLIP" appendContent="false" fieldMapping
  <tag name="PTEST.MOS" value="PTEST" fieldMapping="true" appendContent="false" embedd
```

```
<mosmap id="CamRobot" elementtype="CLIP">
  <fieldmapping fieldname="itemEdDur" fieldtype="FIELDS" mospath="//itemEdDur" valuetype="TIMECODE" mask="ff" />
  <fieldmapping fieldname="objid" fieldtype="FIELDS" mospath="//objID" valuetype="TEXT" />
</mosmap>
```

c. Load the rundown from the Mosart UI and verify that details from the NRCS are passed to the Mosart item:



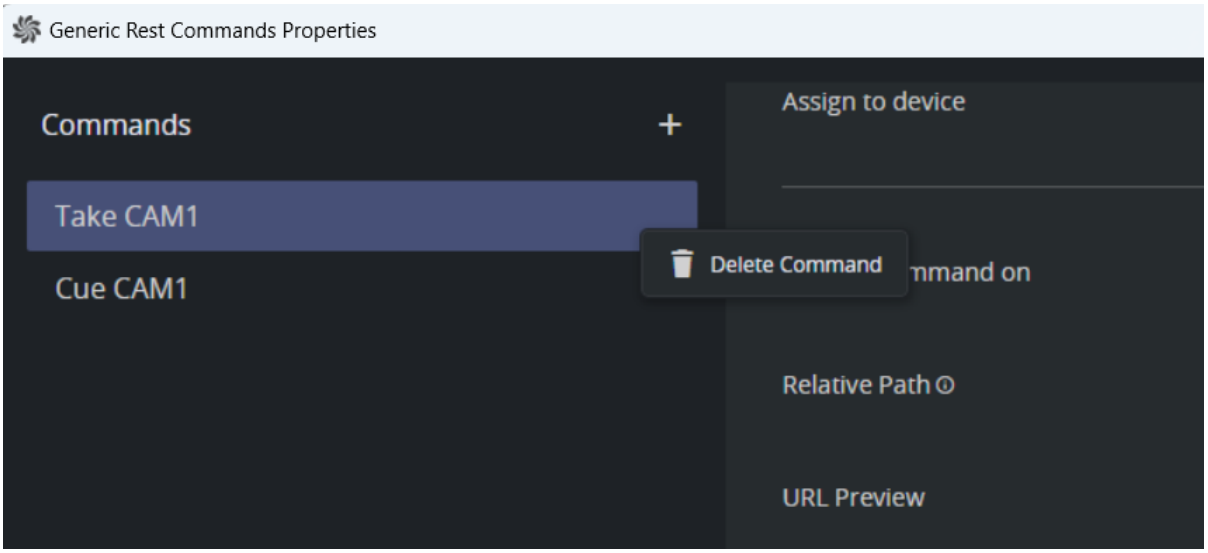
d. Open the ACCESSORIES template and add a Generic REST command with placeholders to resolve to the fields *itemEdDur* and *objid* added on the Accessory item.



Command Management

To remove a configured command

- Click on the three dots to the right of a command row in the list of commands and select **Remove command**.

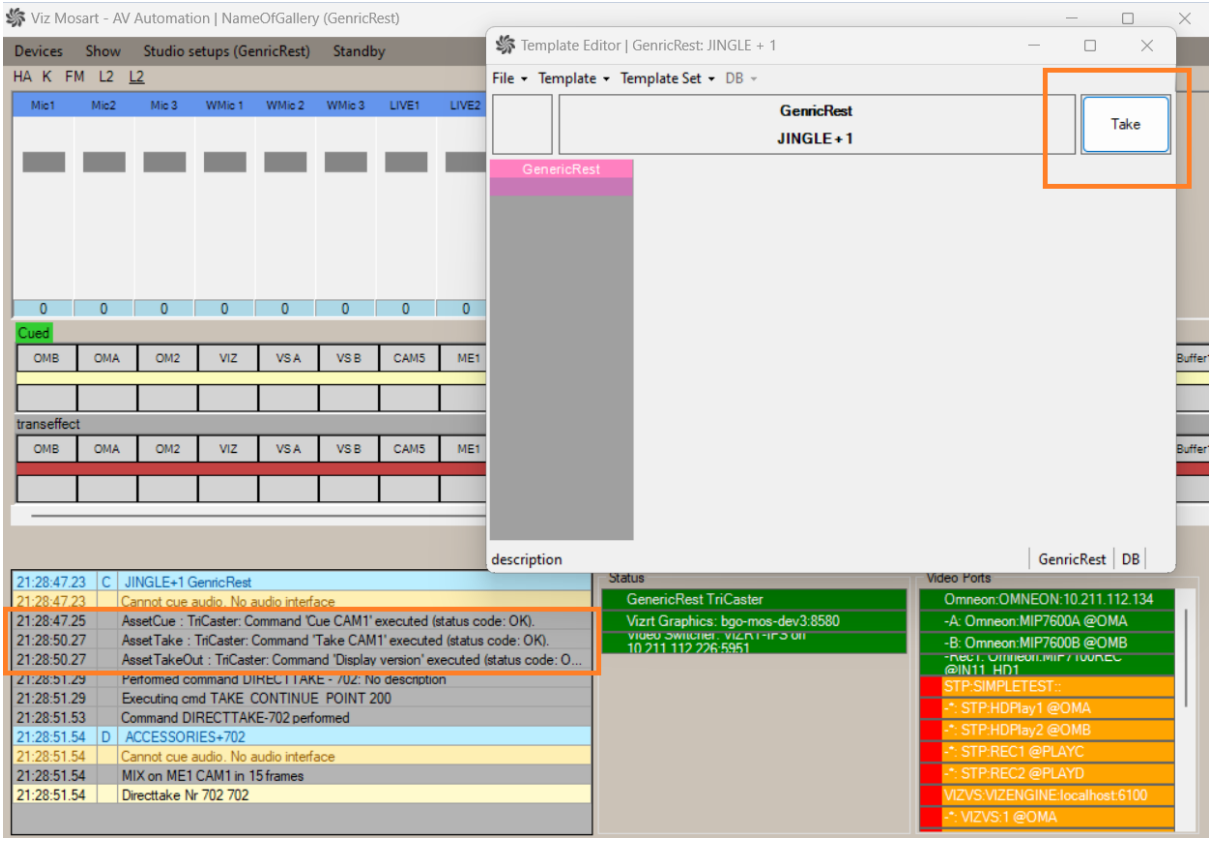


Troubleshooting

Here are some steps to take if results are not as expected.

To verify the execution of a REST command

1. Verify the execution of REST commands configured in a template by running the template using the **Take** button.
2. Check the lower-left **Log** area of AV Automation for execution confirmation from the Mosart server.



3. Confirm the expected outcome on the target device.

If the command failed to execute, verify the Mosart logs for more information.

To verify a REST command is correctly configured

- Copy the URL from the field **URL Preview** and execute the command using a browser or a REST client application like *Postman*.

9.3.9 Mosart Templates Feedback to the NRCS

You can send Mosart templates as MOS objects to an NRCS. The MOS object can then be used by an NRCS operator to insert the detailed Viz Mosart item (rundown event) as a program cue into their rundown (some users call this a “brick system”).

This is an alternative to the Mosart ActiveX, and is the basis for the Viz Mosart web-based application, **NRCS Plugin**.

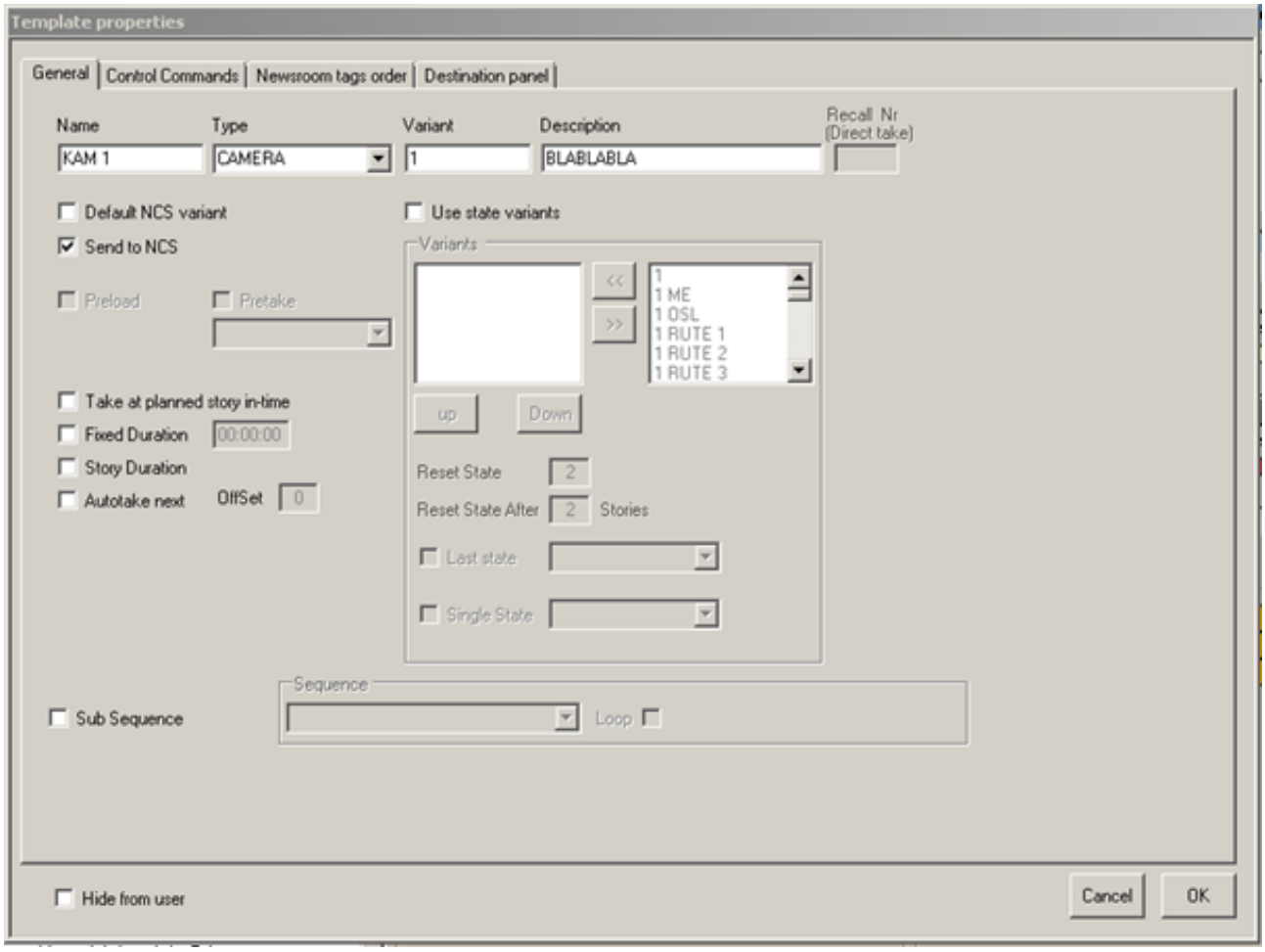
In addition to MOS objects that contain Viz Mosart primary template details, a special MOS object can be sent, containing *lower third* information.

- Template Properties Menu
- Working with Templates as MOS Objects
- Sending Viz Mosart Template Content to the NRCS
- MOS Object Containing Lower third Information
- Usage Notes
- Hierarchical Template Support
- Detailed Description of MOS Objects

Template Properties Menu

You can view and modify the properties of a selected template.

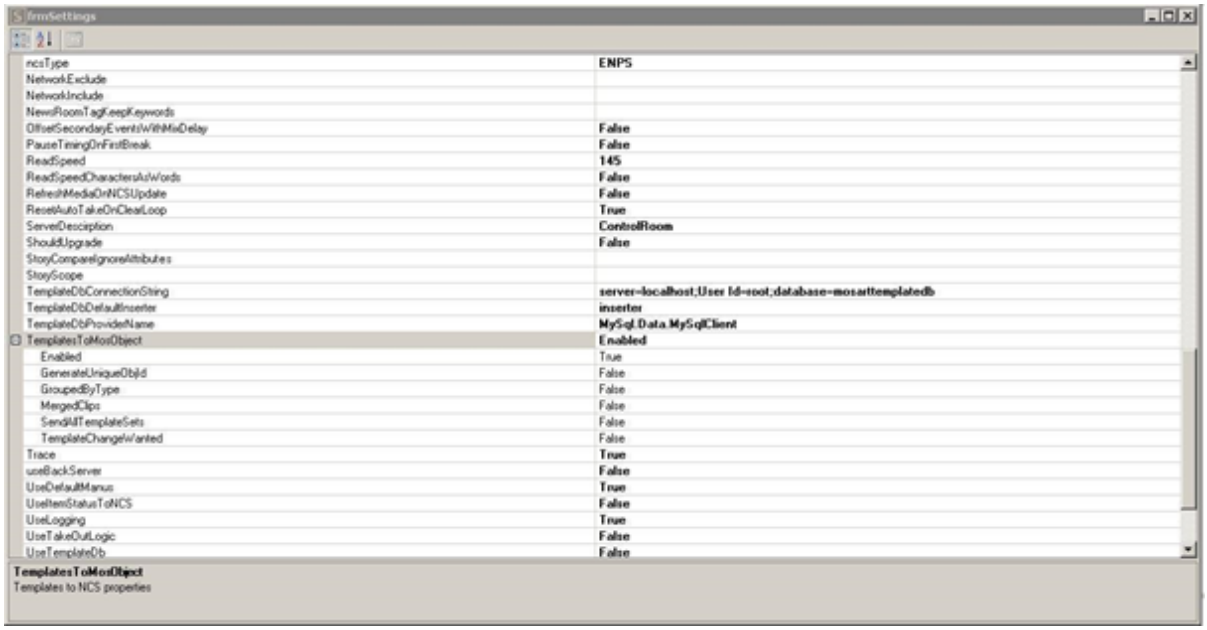
In AV Automation, select **Devices > Template Editor > Template > Properties**.



Working with Templates as MOS Objects

Enabling the Template-to-MOS Object Feature

You enable the features described in this section from the **Viz Mosart Administrator** (MOS version) **Settings** menu.



Settings Parameters

Value	Description
TemplateToMosObject	Used to send templates to NCS through MOS communication. *Enabled*: Enables sending templates to NRCS through MOS communication Default: *Enbaled*
GenerateUniqueObjectId	Some NRCSs (for example, OpenMedia) use **objId** to identify a template. *True:* The MOS object **objId** is sent to the NRCS as the channel template ID , defined as **GUID**. False: When **GenerateUniqueObjectId** is false and - **GroupedByType** (see below) is *false*, **objId** is set to the value of the *TemplateSet-Type-Variant*. - **GroupedByType** is *true*, **objId** is set to the value of the template *type*. **Note**: **objId** is unique among the templates from the *same* gallery. If the same channeltemplates file is used from two different galleries, MOS **objId**s are no longer unique.

Value	Description
GroupedByType	Use to send all template types as *one* MOS object, with the variants embedded in the objects from the default template set. • The **SendAllTemplateSets** property must be set to *false*. • The default template set must first be defined in • AV Automation: **Devices > Template Editor > Template Set > Set As Default**. • If the **SendAllTemplateSets** property is set to *true*, a MOS object for each template type (with the variants embedded in the object) is sent for each template set. • If no default template set is set, then *no* template will be sent to the NRCS. *True*: **objId** is set to the type of the template. *False*: Send all template variants as *separate* MOS objects. Default: *True*
MergedClips	Use this option to merge only the clips from the default template set, regardless of the value of **SendAllTemplateSets**. *True*: Send all PACKAGE and VOICEOVER templates as a collection of type "CLIP" to the NCS using the NEWT or UPDT commands in the Manus Admin console. This property has to be used together with **GroupedByType** (see above). • The default template set must first be defined in AV Automation: **Devices > Template Editor > Template Set > Set As Default**. • If no default template set is set, then *no* template will be sent to the NRCS. Default: *true* See Appendix section Structure of MOS objects sent to NRCS for details about the MOS object structure when merging clips.
SendAllTemplateSets	Use to send templates from all template sets to the NRCS. • The default template set must first be defined in AV Automation: **Devices > Template Editor > Template Set > Set As Default**. *False*: Send only the default template set. Default: *False*
TemplateChangeWanted	Use to decide whether templates changed are updated in the NRCS. • When a template is changed or deleted on the Viz Mosart side, the NRCS will only receive an update if - **mosReqObjList/mosReqAll** (from the NRCS) or - **NEWT** or **UPDT** (from the Viz Mosart Manus console) has been executed at least once since Viz Mosart Manus console was started. • Only a template checked in the field **Send to NCS** (in the **Template properties** menu), is sent to the NRCS. *True*: Always receive updates.

Sending Viz Mosart Template Content to the NRCS

There are two ways to send Mosart templates to the NRCS:

1. Requesting from the NRCS using commands like
   a. mosReqObjList
   b. mosReqAll
   c. mosReqObj <objId>
2. From the Viz Mosart Manus console, using commands like:
   a. NEWT: Send the Viz Mosart templates to the NRCS as MOS objects with status *NEW*.
   b. UPDT: Send the Viz Mosart templates to the NRCS as MOS objects with status *UPDATED*.
   c. DELT: Send the Viz Mosart templates to the NRCS as MOS objects with status *DELETED* (meaning that the templates will be deleted).

- To see samples of the corresponding MOS XML code, see section *Detailed Description of MOS Objects* below.

> **Note**: You can also use the new Viz Mosart NRCS Plugin to quickly drag and drop the underlying MOS object details of a selected Viz Mosart template, directly into an NRCS rundown.

Caching MOS Objects for Sending to the NRCS

All the MOS objects corresponding to the templates sent to NRCS are saved in a cache file

- MosTemplateCache.xml under %localappdata%\temp\Mosart Medialab\MOS State.

This file is created or updated when Viz Mosart Administrator (MOS version) is started.

Troubleshooting

This file is useful when troubleshooting, enabling you to verify exactly what MOS objects have been sent to the NRCS.

You can delete it, then perform a fresh request of the templates, using a **NEWT** command.

Ignoring the *templateset* Attribute for a Viz Mosart Item

When sending *all* Mosart templates to the NRCS (**SendAllTemplateSets** is *true*), an attribute **templateset** is set inside the MOS object.
This attribute is used internally to identify which Viz Mosart template set the type and template type (variant) shall be used.

When a rundown is later loaded in Viz Mosart, the NRCS returns the MOS object with this **templateset** attribute set.

Example

1. You change the studio setup in the Viz Mosart UI by selecting another Template Set (the *active* template set) which contains some of the variants used in the previous template set, but with different properties (e.g same CAMERA 1, but with different crosspoints)
2. The item from the template set referred to by the attribute **templateset** will be executed, instead of the item from the new, *active* template set.
3. To get round this, you can instruct Viz Mosart to ignore the **templateset** attribute (by setting it to an empty value) from a MOS object coming from NRCS.
   Now the Viz Mosart item from the *active* template set will be executed.

a. Do this by adding a **mosid** tag with value *MOSART* in newsroomsettings.xml with a keyword set to *IgnoreTemplateSet*:
<tag name="some.mosid" value="MOSART" appendContent="false" fieldMapping="false" keywords="IgnoreTemplateSet=true"/>
or
<tag name="some.mosid" value="MOSART" appendContent="false" fieldMapping="false" keywords="IgnoreTemplateSet"/>

MOS Object Containing Lower third Information

- For *graphics*, please refer to section Sending Lower Third Graphics MOS Object to an NRCS.
- The corresponding MOS XML code is presented in Detailed Description of MOS Objects.

Usage Notes

Automatic Sending of Viz Mosart Templates to the NRCS

Templates are automatically sent to the NRCS whenever Manus Admin on the Viz Mosart server is started/restarted if **Template feedback to NCS** is enabled in the Manus Admin settings and **TemplateChangeWanted** is set to *true*. In this case, Viz Mosart will send:

- All templates from all sets if **SendAllTemplates** is *true*.
- The templates from default template set, if **SendAllTemplates** is *false* and a template set has been defined as default.
- No templates, if **SendAllTemplates** is *false* and no template has been defined as default.

See the Settings Parameters table above.

Creating and modifying Templates

- When a new template is created or an existing template is changed
  - The template set attribute is defined in the MOS object
  - <objGroup> is set to the template set that the template belongs to.
  - On the NRCS side, the template is stored in a folder named after the template set.
- When a template is changed or deleted, the NRCS will only receive this update when either the commands
  - mosReqObjList/mosReqAll (from the NRCS) or
  - NEWT or UPDT (from Viz Mosart Manus console)
  are executed at least once since Viz Mosart Manus Admin was started.
- To always receive these updates, set **TemplateChangeWanted** to *true*.

  ⚠ **Note**: When **GroupedByType** is set to *true*, deletion of templates is not handled.

  - Execute UPDT in the Viz Mosart Administrator console or request the templates from the NRCS for update

Hierarchical Template Support

The template feedback to NRCS functionality partially integrates with Hierarchical Templates, supporting the two operations

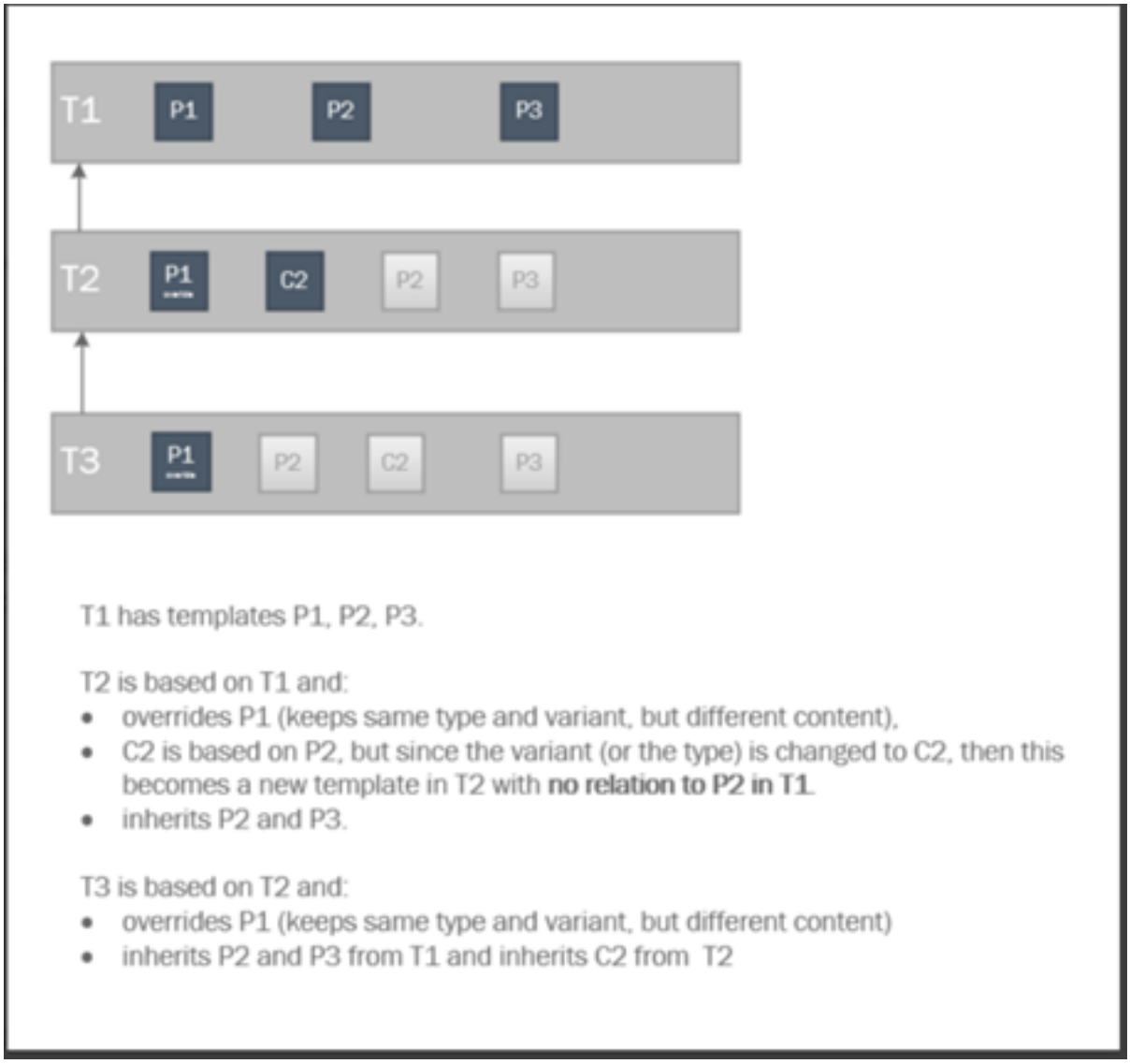
1. Requesting a single template set
2. Requesting *all* template sets

from a gallery with hierarchical templates.

> **Note**: The management of a template set (NEW, UPDATE, DELETE a template) is *not* supported.

Example

Assume the following template sets:



- If template set T1 is requested, then P1, P2 and P3 are sent.
- If template set T2 is requested, then P1 and C2 from T2 and P2 and P3 from T1 are sent.
- If template set T3 is requested, then P1 from T3, C2 from T2 and P2 and P3 from T1 are sent.

Detailed Description of MOS Objects

See the Appendix for code-level descriptions of

- MOS Objects with Mosart Templates Grouped by Type
- MOS Object with Mosart Templates not Grouped by Type
- MOS Object with Clip Templates Merged in a Mosart Template
- MOS Object with Lowerthird Information
- Octopus

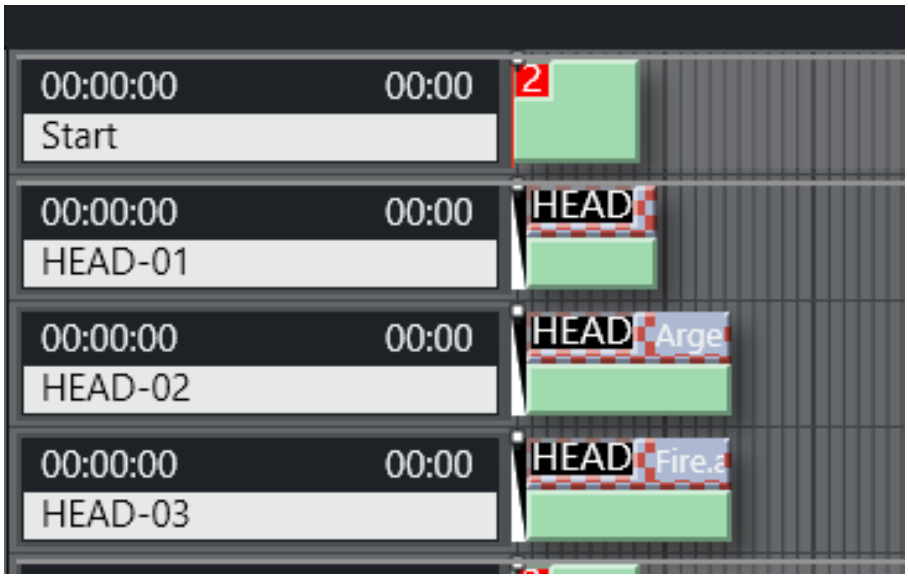
9.4 Other Template Functionality

- State Variants
- Dynamic ME Allocation
- Template Editor Password

9.4.1 Additional Template Features

State Variants

A story sequence comprises items based on templates. With the help of state-based variants, a single template can be used for *all* stories. For example, the figure below shows, after the first story, a sequence of three stories where each story contains a single story item, and each story is using the *same* Viz Mosart template (VOICEOVER+HEAD). State variance enables a different template variant to be used depending upon the story's position in the story sequence.



For example, the first story in a sequence can behave differently to the remaining stories. Another example is when combining a sequence with music, so that each story in the sequence has a specific length, synchronized with the music.

In both examples, is it necessary to invoke different templates depending upon the position of the story in the sequence. You can make it simple for the user to add their stories by just referring to the master template (HEAD in this example).

> **Note:** Template variance only works on *Story* level, meaning that only *one* story item / template should be assigned per story.

Example of Using State Variants

Continuing with the headlines example presented above, the templates can be placed into a sequence.

1. The template called HEAD is the *master template*, and the first story with a HEAD variant uses this template.
2. The second story with a HEAD variant uses the template called HEAD2,
3. HEAD3 is used for the third story.
   In this way, the only command (template variant) used in the newsroom system is HEAD, regardless of whether it is the first, second or third.
4. You can now change the order of the headlines without any modification to the script, making it very quick to do last minute changes.
   In the same way, you can also define which template to use if there is only one HEAD, or if the last HEAD template should act in a specific way.

To Configure a Template State Variant

Configuring template state variance is done in the template editor using the **Template Properties** dialog.

1. Select any template as the *master template* which will be used as the first template in the sequence.

2. Add additional variants in the sequence using the **<<** and **>>** arrow buttons.
   For a particular template type, it is only possible to select variants of the same template type. The same template variant may be used multiple times.
3. Use the **Up** and **Down** buttons to shuffle the variant order.

The following properties provide additional control over which template variant will be used:

- **Reset State**: Index number of the template variant that will continue, if the number of stories in a sequence is larger than the specified number of variants. The index *0* refers to the *first* variant (i.e. the master template itself).
- **Reset State After**: The number of stories not belong to the sequence, that can be run before the new sequence is started. This allows the variance sequence to be kept, even when there are stories within the sequence that are not part of the sequence itself. See the example below.
- **Last state**: Special variant to be used for the *last* story in the story sequence.
- **Single State**: To be used as the last state when there is only one story in the sequence.

Note: When using **Single State**, the property **Last state** must be defined.

Example - Reset State After

This example continues with the story sequence presented above with the three stories all using the HEAD template.
The HEAD template variant has a variance sequence of *three* item: HEAD, HEAD-2 and HEAD-3. Executing the story sequence then invokes the template variants "HEAD", "HEAD-2" and "HEAD-3" for the three stories respectively.

If you now add another story sequence, for example HEAD, HEAD, *OTHER-STORY*, HEAD, with **Reset State After** defined, then executing this story sequence invokes template variants depending upon the value of **Reset State After**:

- **Reset State After <= 1**: HEAD, HEAD-2, OTHER-STORY, HEAD
  The sequence is restarted since **Reset State After** is less than or equal to 1 (story).
- **Reset State After > 1**: HEAD, HEAD-2, OTHER-STORY, HEAD-3
  The sequence is continued since the **Reset State After** is larger than 1 (story).

Dynamic ME Allocation

- To be able to use dynamic allocation, the vision mixer must have at least two mix effect banks in addition to your program/preset bank.
- Viz Mosart uses ME1 and ME2 for the dynamic allocation.

Note: Programming of the emem registers on the vision mixer vary between the various models on the market. Contact Viz Mosart support for instructions on how to prepare your switcher to work with Viz Mosart's dynamic ME allocation.

- When ME's are included in a template with crosspoints or an emem recall, it's possible to do a *dynamic selection* of the ME, by choosing M_RIP or M_OTH.
  Dynamic allocation can be useful in situations where you want an effect to load on an ME not in use, or to set a crosspoint on the ME currently in use.
- To allocate a "new" ME in your template, then choose M_RIP (ME ripple). This loads the effect or set crosspoint on the ME that has the status of "not in use".

Copyright © 2026 Vizrt

When this template goes On Air, the status of the ME will change to "in use". The next template that uses M_RIP will then load on the other ME.

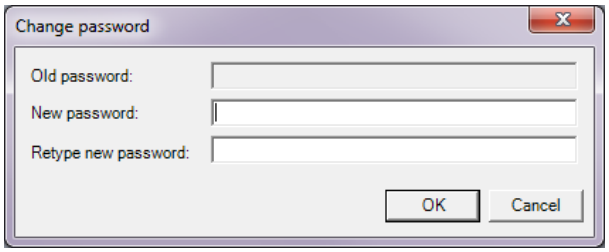
- Conversely, use M_OTH (ME other). Using this in a template loads the effect or set crosspoint on the ME with the status "in use", and does *not* cause a change of status between the two ME's.
  This means that if the next template also uses M_OTH, it addresses the same ME as the previous.

Template Editor Password

Password protection can be enabled for the Template Editor. When enabled, the user is prompted for the password when opening the Template Editor.
By default, *no* protection is enabled, and the user is not asked for a password before opening the Template Editor.

To Enable or Change a Template Editor Password

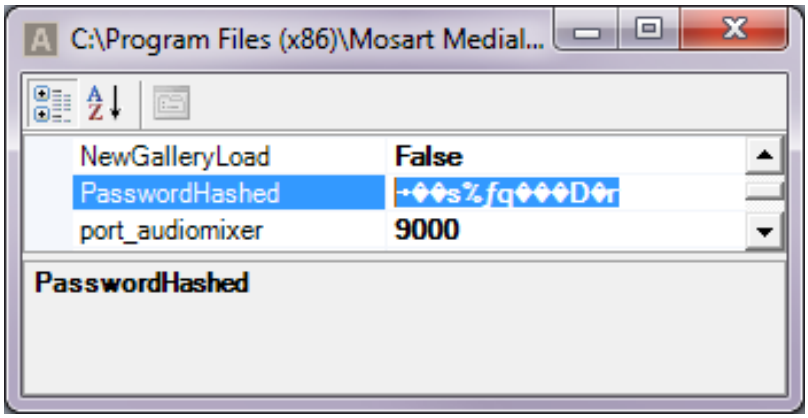
1. Set focus on AV Automation.
2. Select **Devices > Change Password**.
3. In the **Change password** dialog box, enter a new password or change an old one.



To Reset a Template Editor Password

If the password is lost, it can be reset.

1. Set focus on AV Automation.
2. Opening AV Automation Settings with **CTRL+SHIFT+S**.
3. Find the PasswordHashed entry, and remove the value.



9.5 Working with Template Sets

This section contains the following topics:

- Template Set Types
  - Standalone
  - Hierarchical
- Template Set Operations
- Template Set Properties
- Template Set Statistics

9.5.1 Template Set Types

A Viz Mosart template and its variants are arranged in a *set*. There are two approaches to working with template sets.

- Standalone
- Hierarchical

Standalone

Traditionally, Viz Mosart template sets are treated individually, with no connection to each other.

For example, a template set is created for each show or program. This is usually combined with automatically selecting the template set through the associated NCS rundown. In addition, startup commands for preparing the studio (lights, sound etc) for running the show are included in the template set.

The copy contains the same templates as the original, and each template set created in this manner can then be changed without affecting the other. When working with the template set for a *new* program, (based on a copy of an existing program's template set), you make small adjustments according to the functionality needed by the program.

When this new template set is created (by copying an existing set), a copy of *all* templates and any nested sets is made, and no references are made between the original and copied templates. Each template set is classified as ‘standalone’.

Disadvantages with this approach

- Many duplicates are created. Normally only a few of the templates in the set need to be adjusted for meeting the needs of a new program.
- With an ever-growing bank of templates being created for each new show, the amount of stored templates in Viz Mosart can exceed system norms. This results in long response times when saving templates and increases the memory footprint of both the Mosart Server and Viz Mosart client workstations.
- Maintenance of templates becomes cumbersome and error prone. When a change is made in a template that is common for all template sets, the same change has to be applied individually to every template set. This is a time consuming task across numerous template sets.

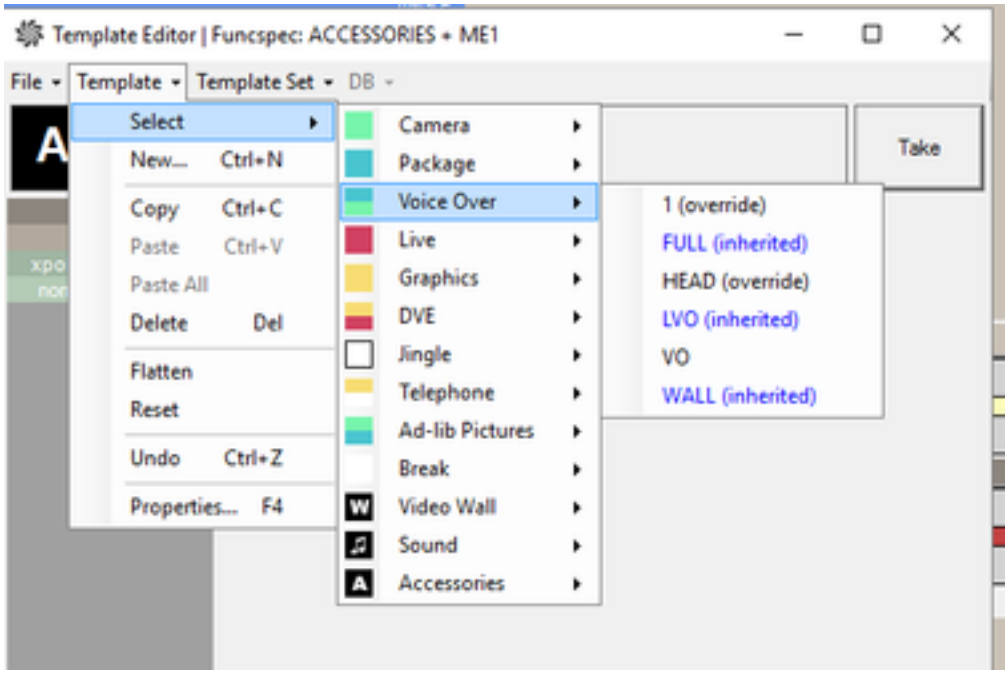
Copyright © 2026 Vizrt

Hierarchical

An alternative approach to standalone is *hierarchical* template sets. This method also enables a new template set to be created, based on the logic of an existing template set. However, to begin with, *no* templates are copied from the existing template set.
Only when a change is made to an existing template, or when a new template created, is a new template created and stored in the new template set.
The hierarchical approach can be adopted with all existing (standalone) template sets, to optimize machine capacity and introduce a more efficient workflow.

Guidelines for this approach

You can see the relationship between a template inherited from one set to another (created using the hierarchic approach), by selecting properties of the template from the newer set.



- **Black letters without any parenthesis**: Operation is defined in the current template set only
- **Black letters with (override)**: Operation is defined in the template set but also exist in the base template set. The current template set definition is used and overrides the implementation in the base
- **Blue letters with (inherited)** : Operation is defined in a base template set and has no implementation in the current template set
- **Red letters**: An empty template, this operation is not defined.
- You can break the relation (inheritance) between two template sets
- You can base a new template set on a template set that was itself created using the hierarchic approach
- You can 'flatten' an inherited template set back to its base (original) template set. This will push any new templates back into the originating template set. After this operation, the new (inheriting) template set is deleted

- When renaming the originating template set, the relationship to any inheriting (based on) template sets still works.

9.5.2 Template Set Operations

To work with a template set, from the AV Automation utility, select **Devices > Template editor**, to open the editor. From here you have several operations:

To Add a Template Set

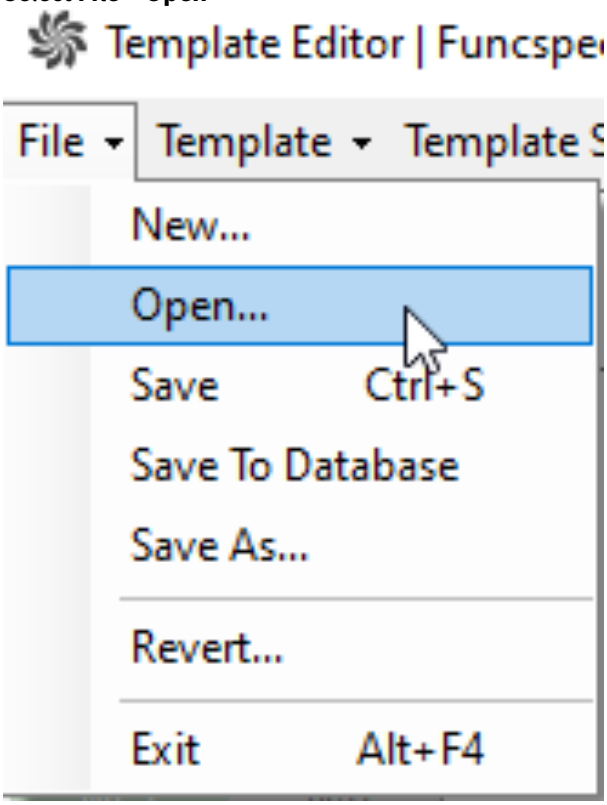
1. Select **Template Set > New**.
2. In the **Name** field, enter the name of the new set.
3. Click **OK**.

File-based Template Sets

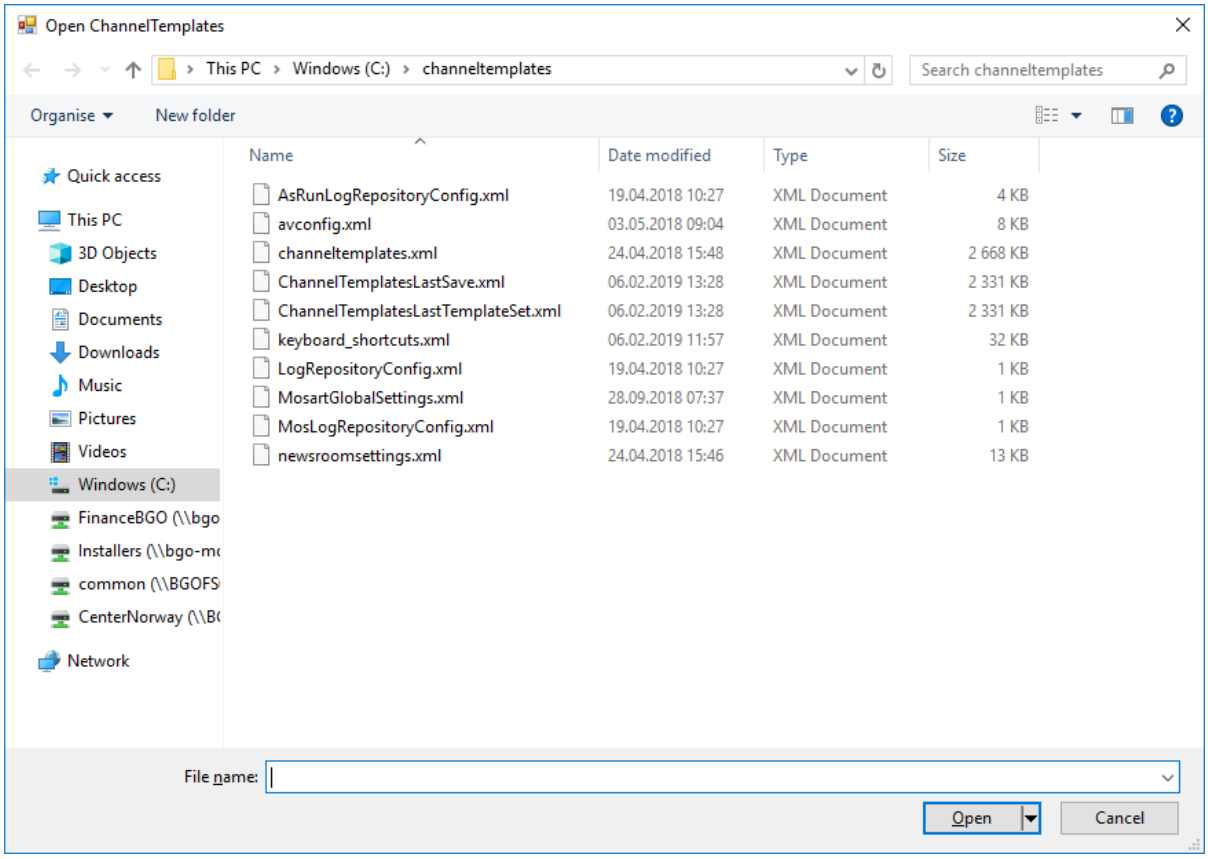
This method can be used *without* the Template Database, for example at installation / show design when trying out several solutions/designs.

You can select a ready-made template set, stored as an XML-file. This presents a convenient way of switching between sets.

1. Select **File > Open**



2. Select the preferred XML-file from the list, and click **Open**.



To Open a Template Set

1. Select an existing template set with **Template set > Select**.
2. The name of the currently opened set is shown in the title of the **Template Editor**.

To Rename a Template Set

1. Select **Template set > Rename** .
2. In the **Name** field, enter a new name for the set.
3. Click **OK**.

To Delete a Template Set

1. Select **Template set > Delete** .

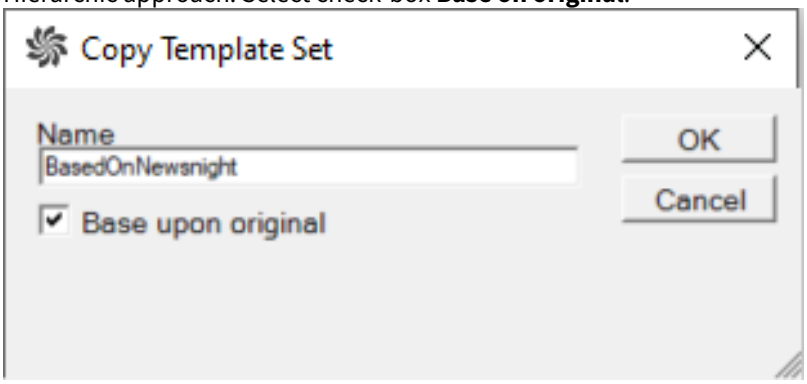
2. In the dialog box, click **Yes** to confirm delete.

> **Note:** You cannot delete a template set that is the origin of any other (hierarchic) template sets. In other words, if a copy of your template set has been made using the hierarchic approach (see To Copy a Template Set below), you cannot delete the originating (base) template set

To Copy a Template Set

Referring to the background discussion above, when you copy a template set, it is more efficient to use a *hierarchical* approach.

1. Select **Template set > Copy**.
2. Enter the name of the new set in the text box.
3. Hierarchic approach: Select check-box **Base on original**.



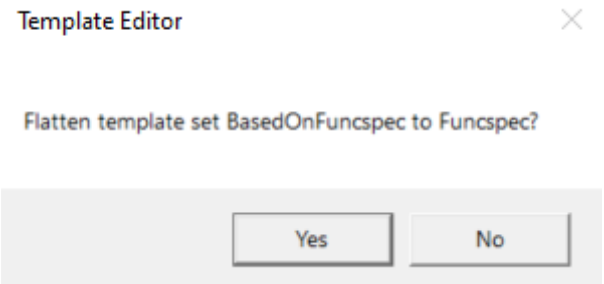
> **Note:** If the **Base on original** check-box is *not* selected a physical copy of the entire template set including all nested templates, is created.

4. Click **OK**.

To Flatten a Template Set

You can push all templates in a new template set (created using the hierarchical approach), back to the originating template set.

1. Select **Template set > Flatten** .
2. A dialog box displays which template set the selected template set will be flattened to.

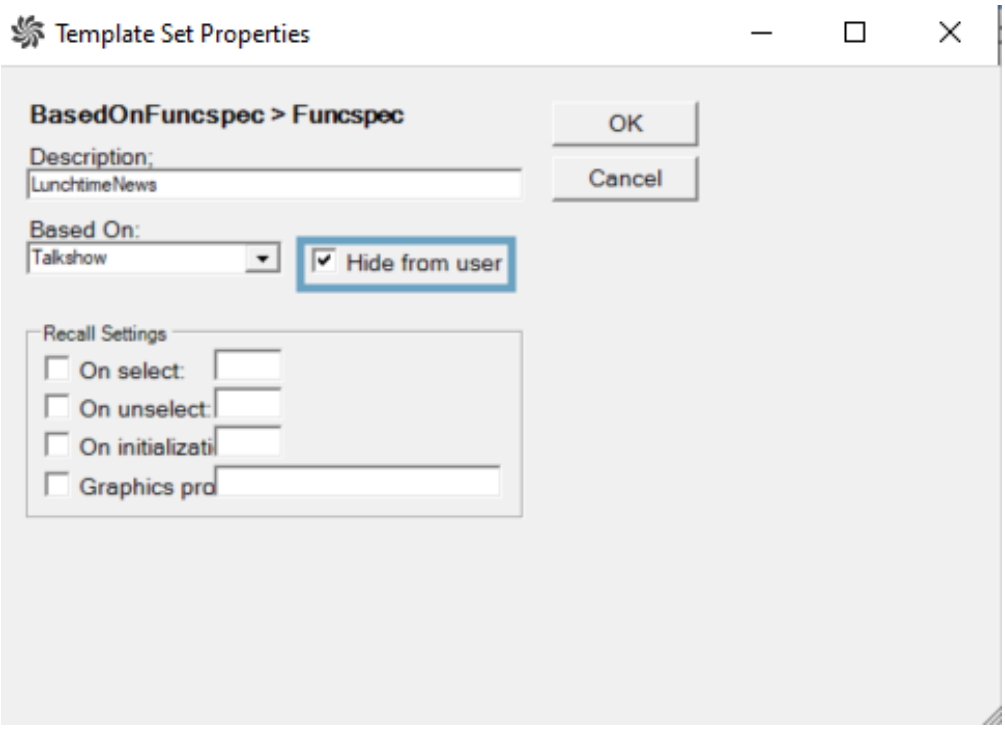


3. Click **OK**.
   All templates in the selected set are now inherited from the base set.

To Hide a Template Set

When a template set is hidden it is unavailable to users of the Viz Mosart GUI and any NCS components (ActiveX) that normally provide user access to select template sets.

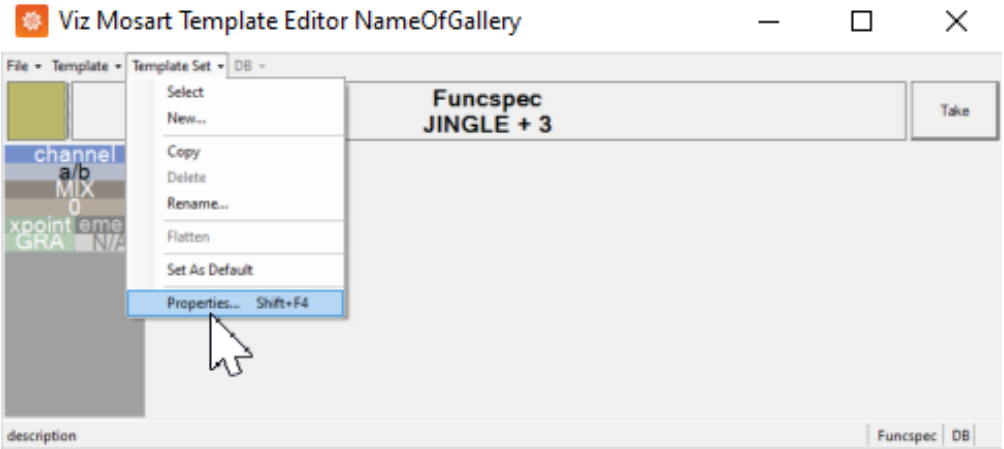
1. Select **Template set > Properties** .
2. Select checkbox **Hide from user**.



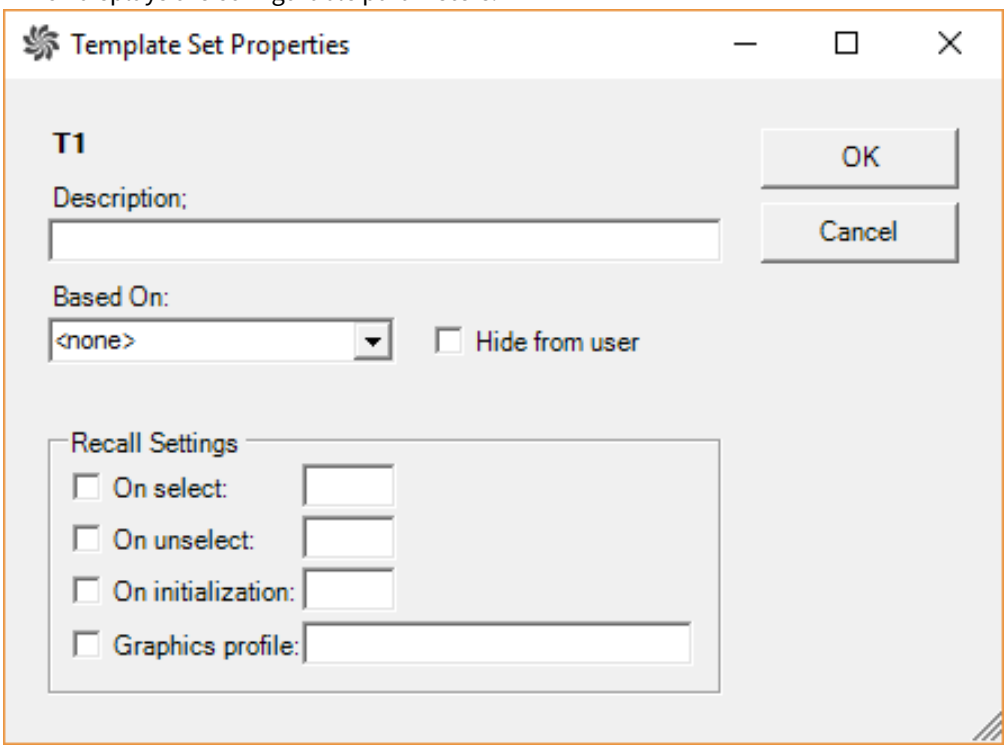
3. Click **OK**.

9.5.3 Template Set Properties

- You can manage some properties of a template set with **Template Set > Properties**.



Which displays the configurable parameters:



Item	Description
Description	The name of the template set.
Based On	Original template set, upon which this template set is a unique variant.
Hide from user	Prevent template set from appearing in user selection menus.
On select	Behavior of recall under this condition.
On unselect	Behavior of recall under this condition.
On initialization	Behavior of recall under this condition.

Item	Description
Graphics profile	(When used) Profile that will be applied to the new template set.

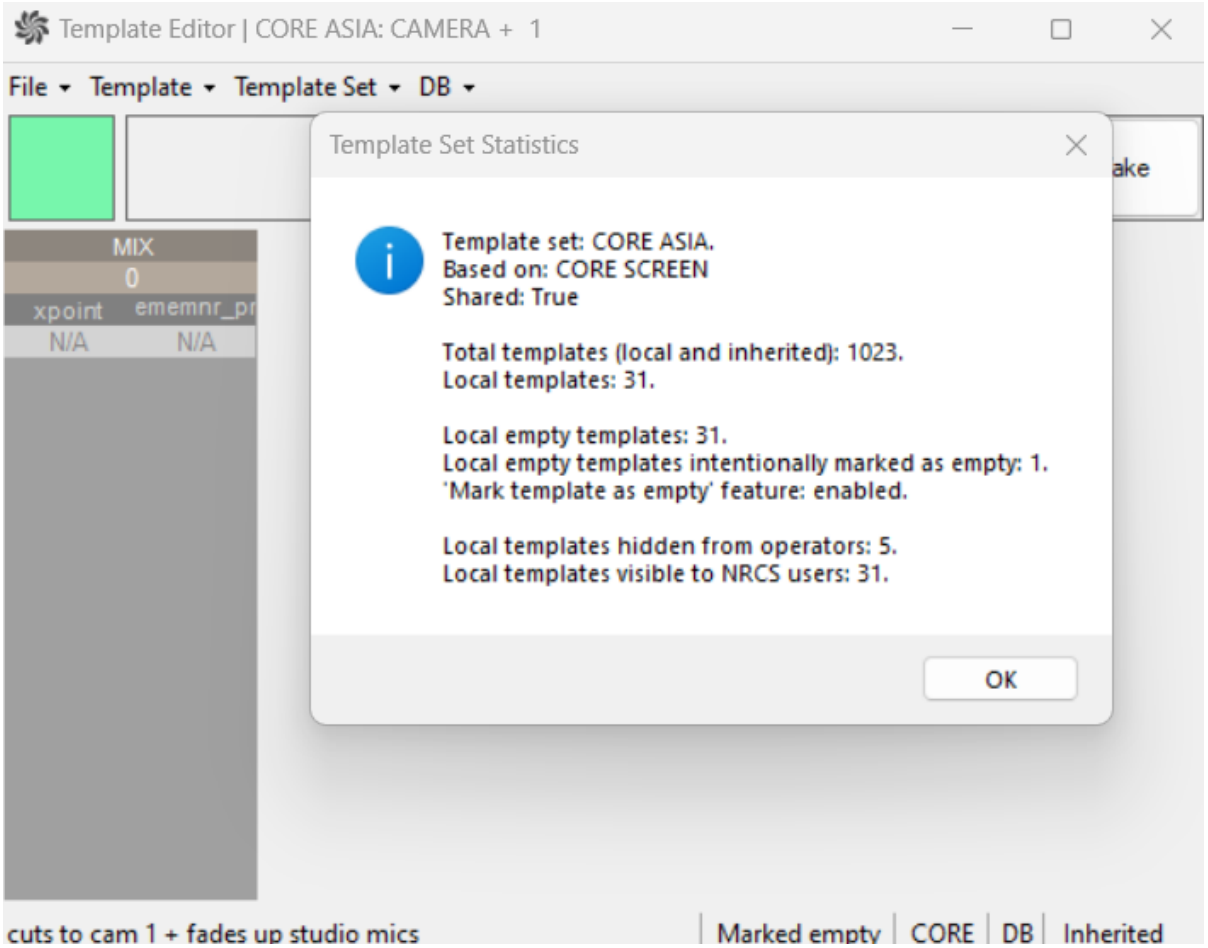
9.5.4 Template Set Statistics

The Statistics menu feature found in **Template Set > Statistics** provides a summary of key details for each template set, helping users quickly assess the configuration and characteristics of templates within the set. Statistics displayed include:

- **Template Set Overview**: Name of the template set, the base template set from where it is derived, and whether the set is shared across multiple galleries.
- **Template Counts**: Total number of templates in the set, distinguishing between local and inherited templates.
- **Empty Template Details**: Number of empty templates in the set, specifying how many are intentionally marked as empty.
  If the "Mark Template as Empty" feature is enabled, this status is confirmed here.

> **Note**: See iNEWS/MOS Settings Editor under Manus Administrator Configuration for **Use 'Mark template as empty' feature** details.

- **Visibility Settings**: Number of templates hidden from operators and those visible to NRCS (newsroom computer system) users.

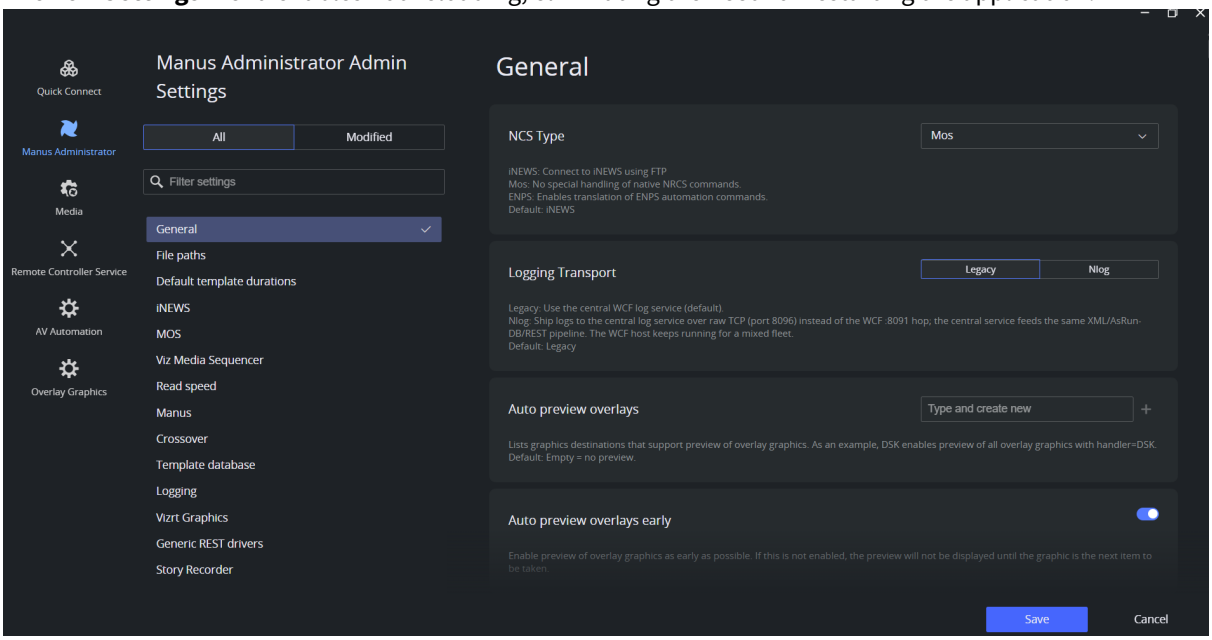

Template Editor | CORE ASIA: CAMERA + 1
File
Template
Template Set
DB
Template Set Statistics
Template set: CORE ASIA.
Based on: CORE SCREEN
Shared: True
Total templates (local and inherited): 1023.
Local templates: 31.
Local empty templates: 31.
Local empty templates intentionally marked as empty: 1.
'Mark template as empty' feature: enabled.
Local templates hidden from operators: 5.
Local templates visible to NRCS users: 31.
OK
MIX
0
xpoint
N/A
N/A
cuts to cam 1 + fades up studio mics
Marked empty
CORE
DB
Inherited

10 Viz Mosart Central Settings

The Viz Mosart **Settings** menu provides a single point to configure all server-side parameters and manage entire operations from a single menu.

- Opening the Settings Menu
- Architecture
- Setup Examples

- In upcoming versions of Viz Mosart, settings for other Mosart Server applications and operations will be transferred to this centralized **Settings** menu, to ease the inventory of the Mosart Server.
  This will be implemented gradually, spanning several Viz Mosart versions.
- For AV Automation, all Device Properties (**Devices > Properties >[Device type** tab**]** will gradually be migrated to this centralized **Settings** menu.
- For examples of simplified operational setups, see Setup Examples below.
- The new **Settings** menu enables hot-reloading, eliminating the need for restarting the application.



10.1 Working with the Settings Menu

The web-based **Settings** menu is accessible from the the combined Manus Administrator that was introduced with Viz Mosart 5.3.

Settings from the legacy Manus applications (MMConsoleAdmin_2007.exe and MMConsoleAdmin_MOS.exe) are consolidated and presented in this improved menu.

- Opening the Settings Menu
- Architecture
- Setup Examples

10.1.1 Opening the Settings Menu

To open the Settings menu

1. Ensure that your installation is using the *combined Manus*, as introduced in Viz Mosart 5.3.
2. Either:
   a. In AV Automation, navigate to **Devices > Settings**, or press **Ctrl**+**Shift**+**P**
   or
   b. In the Manus Administrator console window, type typing *settings.*

10.1.2 Architecture

- Settings are stored in the Mosart Server's local file system, specifically under C:\ProgramData\Mosart Medialab\ConfigurationFiles\manus_settings.json.
- The new settings editor uses Microsoft Edge WebView2 as the browser.
  Please refer to the Prerequisites page during installation for more information.

10.1.3 Setup Examples

The following operations/configurations, managed from the **Settings** menu are presented in detail:

- Generic REST Drivers
- TriCaster
- Settings Editor iNews
- Settings Editor MOS
- Vizrt Graphics Settings

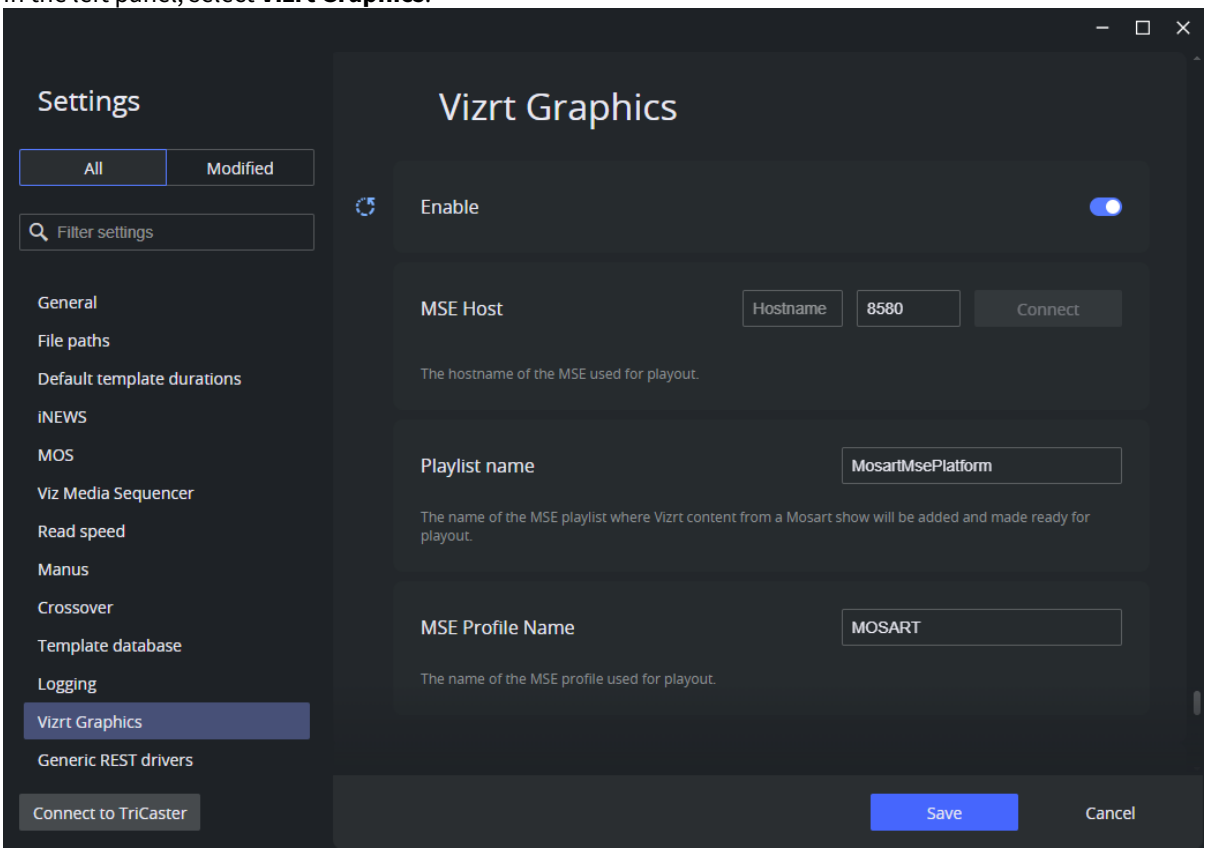
10.2 Vizrt Graphic Settings

For Vizrt graphics (Viz Arc AR elements and limited support for overlays, full screens and clips) setup and workflow is simplified with a single driver that communicates with Vizrt's central media broker, the Media Sequencer (MSE). MSE communicates with AV Automation.

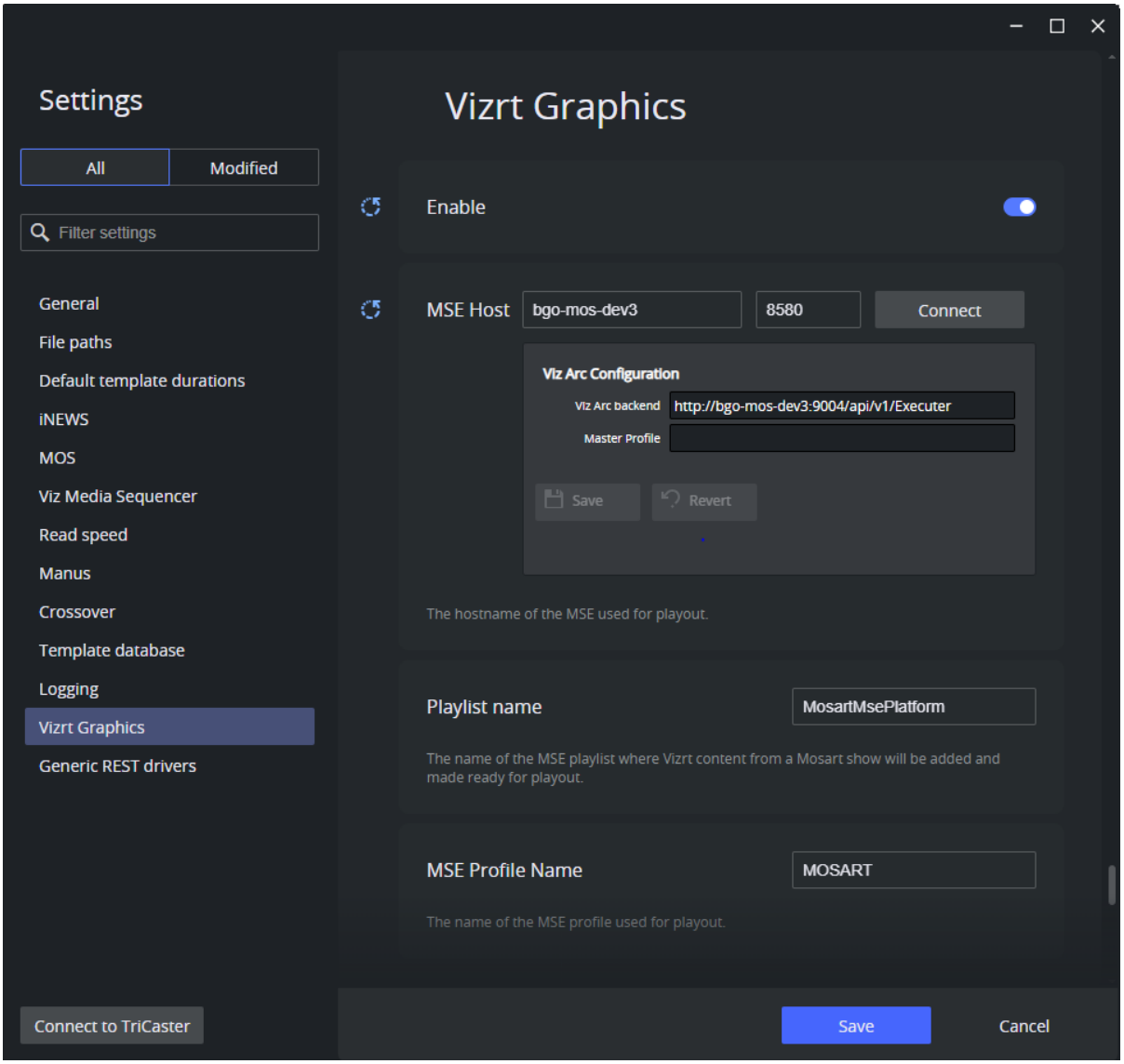
Connection to the Vizrt Graphics driver is configured through the new web-based **Settings** menu.

10.2.1 To configure Vizrt graphics

1. Open the **Settings** menu, as described in Viz Mosart Central Settings.
2. In the left panel, select **Vizrt Graphics**.



3. In the panel to the right, select **Enable**.
4. In the MSE Host section provide:
   a. Host name
   b. Port number (default 8580).
5. Click **Connect**.
   On successful connection to MSE, a page served by Media Sequencer is displayed, just below the host and port fields.
6. In the section **Viz Arc Configuration**:



a. Enter a value for **Viz Arc backend**, as a URL, in the form: *http://<Viz Arc REST service hostname>:<Viz Arc REST service port>/api/v1/Executer*. (default Viz Arc REST service port is *9004*. See the Viz Arc User Guide).
b. The **Master profile** is defined in Viz Arc. Here is an example for Viz Arc 1.8.
A *profile* is used to configure the different groups (channels) of either the *Viz* or *Unreal Engines* available for user control.

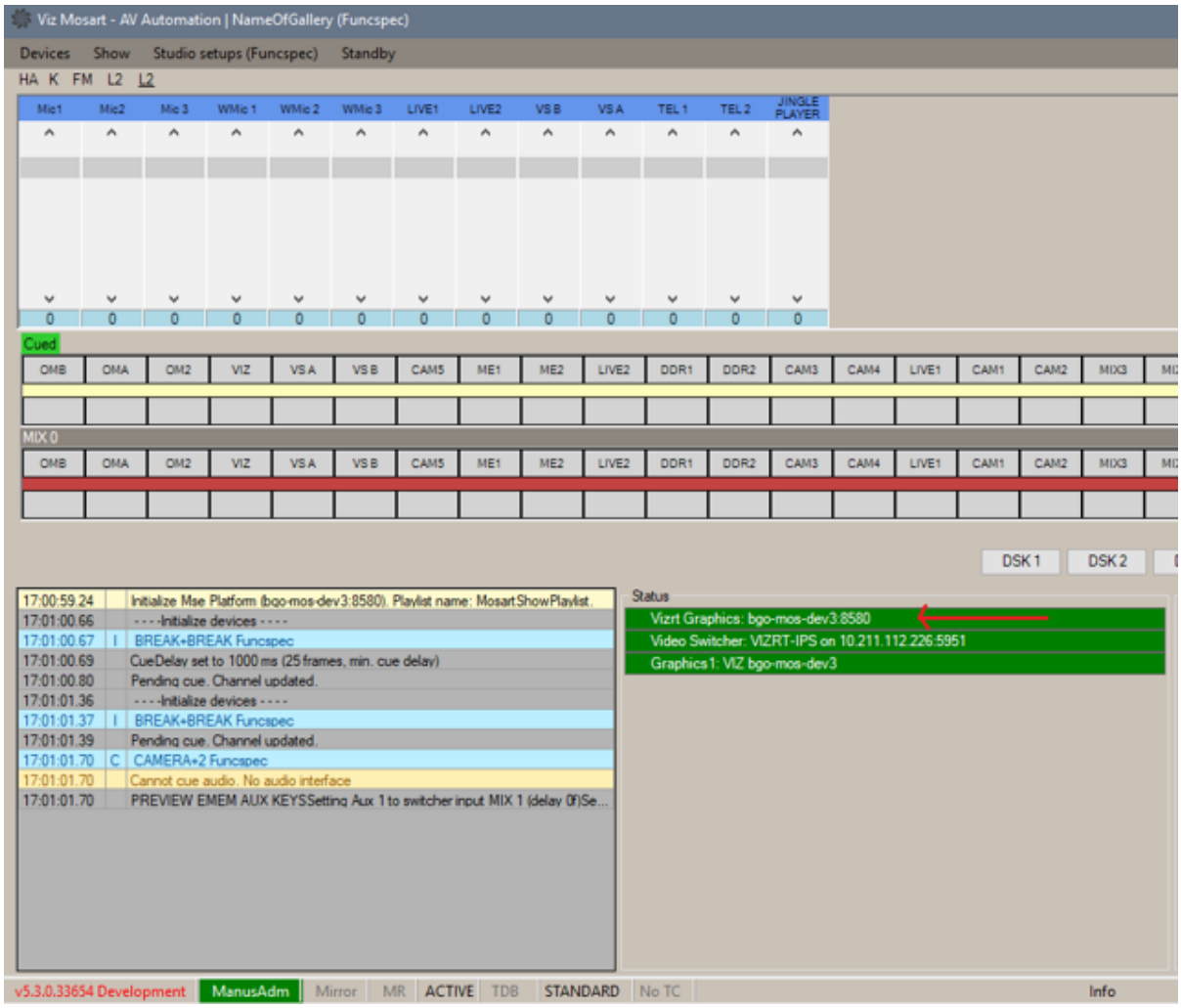
7. Click **Save** to save to the MSE (or **Revert** to undo your changes).

> ⚠ **Note**: These settings are saved on the Media Sequencer side (*not* Viz Mosart)

8. Enter a value for the MSE **Playlist name**.
This is where the Viz content playout elements added in a Viz Mosart show, planned in an NRCS, will be added to be handled through the Vizrt Graphics driver.
This value is *case sensitive*. Default: *MosartMsePlaylist*

9. Enter the **MSE Profile name**.
   This is used for playing out elements. The selected profile must first be *created* and *activated*, using a profile editor available in either Viz Trio, Viz Pilot (Director) or Viz Multiplay.
   This value is *case sensitive*. Default: *MOSART.*
10. Click **Save** to apply the setting.
    There is no need to restart the Manus Administrator.

With the settings applied, the MSE connection status that employs the *Vizrt Graphics* driver is shown in Viz Mosart AV Automation's **Status** panel (see red arrow):



10.3 Generic REST Drivers

Viz Mosart offers an easy to create, generic driver that sends REST commands to an external device. Features for these drivers can be controlled from a standard Mosart template, as described in section Controlling a REST-based Driver.

Key features of Viz Mosart Generic REST drivers:

- **Device configuration**: Configure up to 50 devices that provide REST API services.
- **Authentication support**: Basic, Digest, API key, OAuth2 (Client credentials and Password grants) and Token login, for a device with its own login endpoint. See Authentication types.
- **Certificate pinning**: Connect to a device with a self-signed certificate by entering its certificate thumbprint, instead of disabling certificate validation.
- **Connection status monitoring and heartbeat**: Monitor the connection status of the device, with an integrated heartbeat mechanism for consistent device status checks.
- **Standby mode**: Set devices in standby mode.
- **Template-based command execution**: Configure and execute commands on cue, take or take-out.
- **HTTP command requests**: GET, POST and PUT HTTP request methods.
- **Integration with newsroom systems**: Use newsroom tags in request commands, so journalists can overwrite values with content from their newsroom system.
- **Directtakes functionality**: Execute Directtakes with REST commands for a configured device from templates or keyboard shortcuts in the Viz Mosart UI.

Out-of-the-box features available with a Generic REST driver:

- **Redundancy** using a reverse proxy such as Caddy or HAProxy.
- **Control a Mosart server from a template** using the Mosart Remote Control REST API instead of using GPI control commands.
- **Control other Mosart servers** from a Mosart server using the Mosart Remote Control REST API.

- Setting up and Connecting to a REST-based Device
- Authentication types
  - Example: a device with its own login endpoint (TokenLogin)
  - Example: calling the Mosart REST API when OIDC is enabled (OAuth2)
- Management of REST-based Devices
- Controlling a REST-based Driver
- Troubleshooting

10.3.1 Setting up and Connecting to a REST-based Device

Tip:
- For studio-tested examples, see section REST Driver Implementation.
- A two minute tutorial video on using the REST API is here.

You construct your REST-based driver from the **Settings** menu in the combined Mosart Administrator utility.

1. Open the **Settings** window. There are two methods:
   a. From the Manus Administrator console, type *settings*.

b. From the AV Automation menu bar, navigate to **Devices > Settings** (**Ctrl+Shift+P**).

2. In the left side of the **Settings** menu, select **Generic REST drivers**.
3. Add a new device by clicking **Add device**, or the plus (**+**) sign next to the **Devices** header. You can add up to 50 devices.
4. Configure the device in the right side panel. The core fields are:

Field	Meaning
Device title	A name that uniquely identifies the device. Default: *Untitled*.
Device ID	The generic REST number shown in the AV Automation Status panel and the Standby lists. Usable with a standby device control command.
Base URL	Mandatory: the scheme (http:// or https://) and the host of the device, optionally with a common resource path. It is the shared part of the URL of every request to the device.
Status URL path	The resource path appended to the Base URL for the connection test and the heartbeat.
Authentication type	How the driver authenticates. See Authentication types below.

> **Info**: Every setting explains itself: hover the information icon next to a field for what it does, how to fill it in, and where the value comes from. The authentication fields appear and disappear with the chosen **Authentication type**, so you only see what the selected type needs.

5. Test the connection by clicking **Test Connection**. If the connection fails, an error message is displayed. See Troubleshooting below.
6. Once the connection is successfully established, *enable* the device by toggling the switch next to the device's name. This initiates an HTTP connection to the device, preparing it to receive commands from Viz Mosart.
7. As necessary, add, configure and enable more devices, then save all settings by clicking **Save**.
8. In the AV Automation UI, check that the enabled devices are displayed in the Status panel. With no heartbeat configured, Mosart checks the connection once and shows green for *OK* or red for *not OK*; with a heartbeat, hovering over the device shows the last check. You can reconnect to the device in AV Automation with **Ctrl+Shift+Z**.

> **Note**: If the device is not enabled, Viz Mosart does *not* transmit any commands to it.

> **Info**: A device might not require authentication for the endpoint used to test the connection, in which case a green test does not prove the credentials. Where authentication matters, point the **Status URL path** at a resource that requires it, so the test and the heartbeat prove the whole chain. For example, the Mosart REST API answers `/api/v1/build` without credentials, but `/api/v1/status` only with them.

10.3.2 Authentication types

Set **Authentication type** to what the device requires; its REST API documentation says which. Each type shows only its own fields, and each field's tooltip carries the detail.

Type	For	What is sent
None (default)	A device requiring no authentication.	Requests as-is.
Basic	Simple username and password authentication.	The credentials, Base64-encoded, on every request.
Digest	Challenge-response username and password authentication.	A digest exchange; the password itself never travels.
ApiKey	A device that expects a fixed key in a request header.	The key on every request, in the configured header (default `X-API-Key`).
OAuth2	A device whose API requires tokens issued by a standard OAuth2 identity provider such as Keycloak or Microsoft Entra ID. A Mosart server with OIDC enabled is such a device. Choose the **Client credentials** grant for new devices, so the device signs in as its own service account; the **Password** grant remains for existing configurations.	A token fetched from the identity provider's token endpoint, sent to the device as `Authorization: Bearer` and renewed automatically before it expires.
Token Login	A device that issues the tokens itself, from its own login endpoint, such as the Imagine SNP. The credential format, token response format, token header and refresh interval are configurable, because bespoke device APIs differ in exactly these places.	A token fetched from the device's login endpoint, sent in the configured header and renewed on the configured interval.

Info: To connect to a device with a self-signed certificate, enter the certificate's SHA-1 or SHA-256 thumbprint in **Device certificate thumbprint**: the connection then trusts exactly that certificate, while an expired certificate or one belonging to a different device is still rejected.

Example: a device with its own login endpoint (TokenLogin)

The Imagine SNP is the typical case: it wants a JSON credential body at `/api/auth`, answers with the bare token string, expects the raw token in the `Authorization` header with no scheme, and its tokens expire after roughly five minutes.

Setting	Value
Base URL	`https://<snp-manager>:9089/`
Status URL path	`api/elements/<snp-ip>/isInitialized`
Authentication type	*TokenLogin*
Authentication Server Url	`https://<snp-manager>:9089/api/auth`
Username / Password	the account defined on the device
Credential format / Token response format / Token header / Token scheme	the defaults: *Json*, *RawToken*, `Authorization`, empty
Token refresh interval	`240` seconds, safely under the five minute token lifetime
Device certificate thumbprint	the thumbprint of the device's self-signed certificate

Example: calling the Mosart REST API when OIDC is enabled (OAuth2)

Enabling OIDC authentication on a Mosart server switches the REST API key off, so a Generic REST device pointed at Mosart's own REST API must sign in as its own service account instead (available from Viz Mosart 5.16):

1. Ask IT to create a client for the device in the identity provider, using the client credentials flow, and to hand back its Client ID and secret. If the device manages named overlays, the client's service account also needs the `mosart-admin` role. See the Security chapter.
2. Configure the device: **Authentication type** *OAuth2*, **Grant type** *ClientCredentials*, **Authentication Server Url** = the identity provider's token endpoint, plus the Client ID and secret.
3. Point **Status URL path** at `api/v1/status`, which requires a valid token, so a green heartbeat proves the whole chain.

> **Info**: A Generic REST device that calls a **third-party** service is unaffected by the Mosart server's own authentication mode. With the API key mode, the older setup keeps working: authentication type *ApiKey*, the key from *Remote Controller Service* > *Security*, header `X-API-Key`.

10.3.3 Management of REST-based Devices

To **remove a device**: hover over it in the device list, right-click the three dots to the right and select **Remove device**.

To **enable, disable or remove all devices**: right-click the three dots next to the **Devices** header and select an operation from the context menu.

10.3.4 Controlling a REST-based Driver

Standard Viz Mosart templates can be used to control a REST-based driver. These show-specific operations are described in section Controlling a REST-based Driver.

10.3.5 Troubleshooting

Symptom	What to check
The connection test fails	The *Base URL* format; whether a *Status URL path* is needed; the authentication values; that AV Automation is operational. The Mosart logs have the detail.
401 errors start a few minutes after connecting	The **Token refresh interval** is longer than the device's token lifetime. Set it to roughly 80% of the lifetime.
401 on the token fetch itself	Wrong credentials, or the wrong **Credential format** for the device.
Certificate error although a thumbprint is configured	A typo in the thumbprint, or the certificate has a problem pinning does not excuse: it is expired, or its host name does not match. The Mosart log names the reason and prints the thumbprints the device actually presented.

Copyright © 2026 Vizrt

10.3.6 REST Driver Implementation

Here follows a growing list of real-life examples of improving the management of a production by employing a Mosart REST Driver API.
Appreciation is expressed to the organizations that kindly shared their details with Vizrt.

Examples of REST Driver Implementation

- Robotics Control

Robotics Control

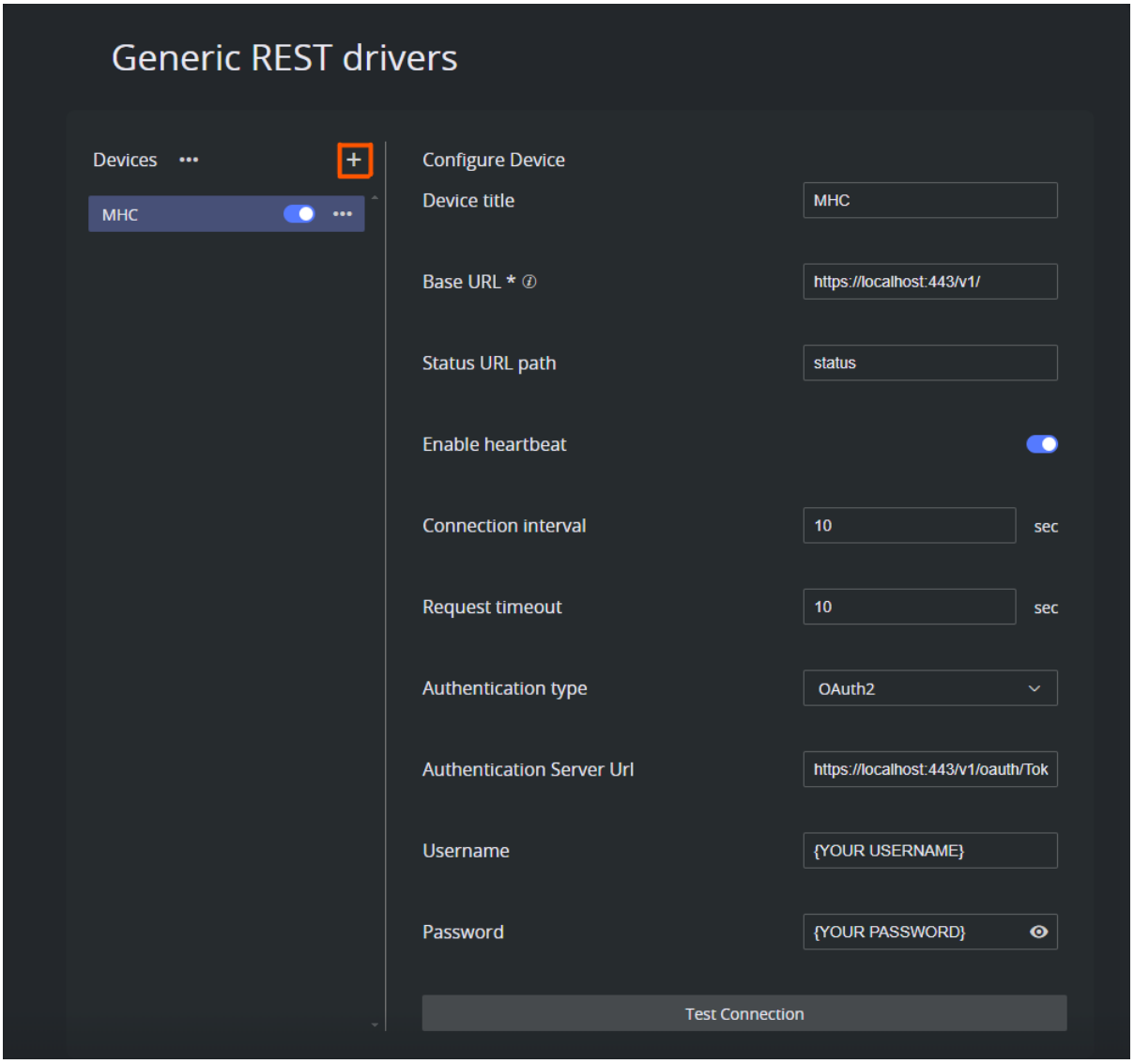
You can control an MRMC Multi Head Controller (MHC) with Viz Mosart's Generic REST API.

Prerequisites

MHC installed on a machine and set up with at least 1 robotic camera.

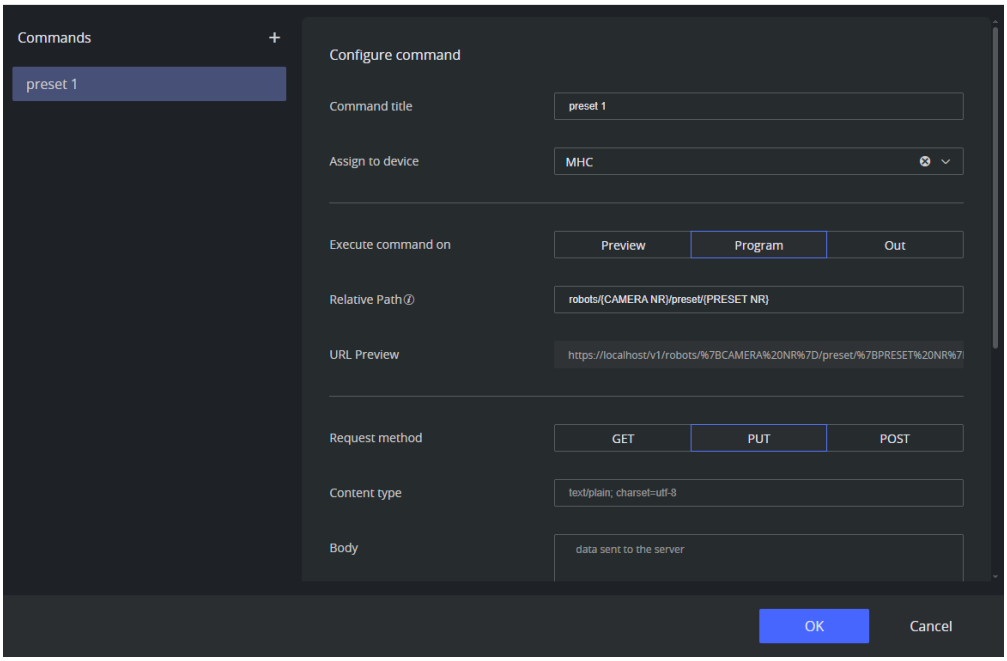
To configure REST API control of a robotic camera

1. Open AV Automation and from the menu bar, navigate to **Devices > Settings**.
2. Scroll down the Settings menu to **Generic REST drivers**.
3. Add a new driver by clicking the **Plus** (**+**) sign.
4. Fill in required details, following the example below:



Note: *Base URL* and *Authentication Server URL* are the machine running MHC, *not* localhost.

5. Click **Save** and close the menu.
6. Open Template Editor.
   (**AV Automation > Devices > Template editor**).
7. Select **File > New** to create a new template.
8. Right-click in the template's dark gray area then **Enable > Generic Rest Commands**.
9. Add a command by clicking the **Plus** (**+**) sign in the left-side **Commands** panel.
10. Fill in required details, following the example below:



Note:

- {CAMERA NR} is the index of the camera you want to use (starting at 0), and {PRESET NR} is the index of the preset you want to load (starting at 0).
  If you want to load preset 3 of camera 2, you would do *robots/1/preset/2.*
- Set **Execute command on** to your requirement, it need not be *Program* as shown in the sample screenshot.

11. Click **OK**.
12. Save the template.
13. Add this template to a Mosart rundown.
    Whenever it is taken, it triggers MHC to move the selected robotic camera to the selected preset position.

11 Frame Accurate System Operations

To ensure reliable operational synchronization between Viz Mosart and the studio equipment that it controls, the system can be calibrated to perform on a frame accurate level of timing precision.
This section introduces:

- Why Frame Accuracy Matters
- Requirements
- Enabling Frame Accuracy Operation
- UTC or Local Time
- Calibrating the System
- Genlock Logging
- Frame Accuracy in Story Recorder
- Limitations

These pages complete the chapter:

- Device Calibration: the tests that measure the latency of each controlled device, and where to enter the values.
- Show Design for Frame Accuracy and Story Recorder: how to build templates and shows that run frame accurately.
- PTP Setup: how to take genlock and timecode from a PTP grandmaster, from Viz Mosart 5.15.0 onward.

11.1 Why Frame Accuracy Matters

There are two areas where frame accuracy in a Viz Mosart controlled system are significant:

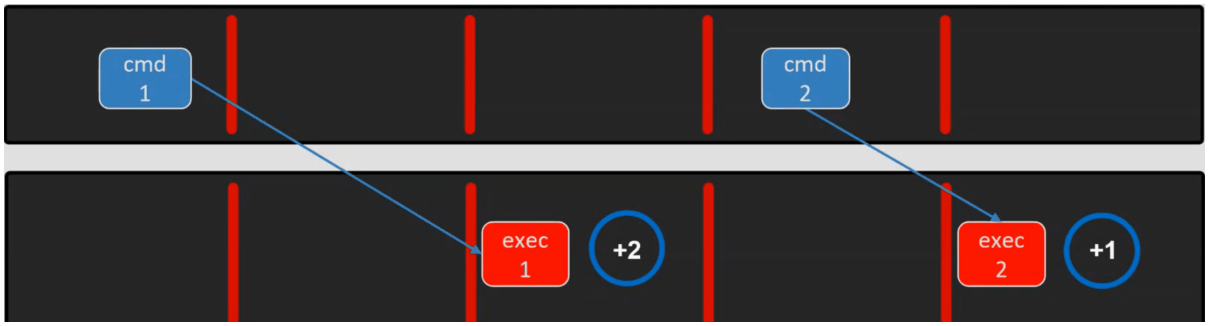
- Predictable Behavior of Equipment
- Latency Compensation.

11.1.1 Predictable Behavior of Equipment

Unless it is known where in the frame the Viz Mosart command is issued, the response from the equipment will always be arbitrary.

In the figure below, the red bars represent the start of a frame. The outgoing Viz Mosart commands (*cmd 1*, *cmd 2*) are now aligned with the start of the frame grid (frame accurate).

Tip: The illustrations in this section are *animated*. If you are viewing a PDF, we recommend the HTML version of this document, the latest can always be found online from here.

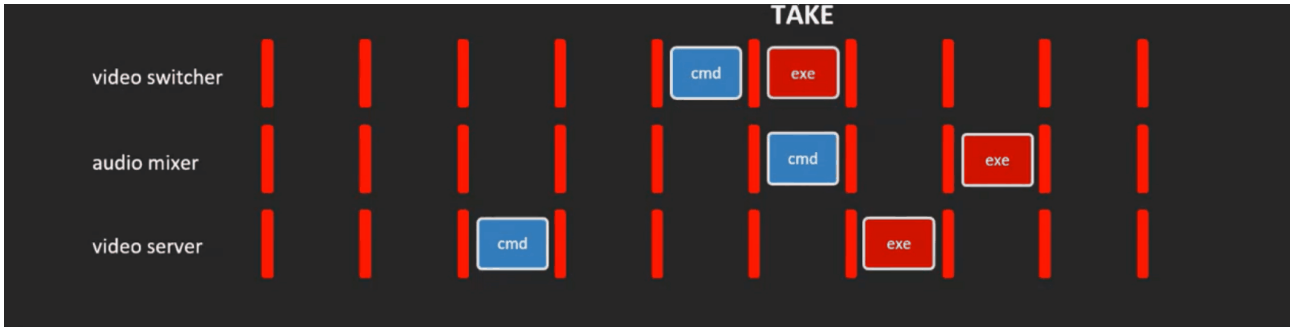


This provides deterministic behavior from the controlled equipment, as the equipment is always given exactly the same amount of time to respond (in the above figure, after *one* frame).
Without frame accuracy, *cmd 1* might be fired already somewhere in the first or second frame above, and the gear will not operate at the anticipated frame, giving unpredictable results.

11.1.2 Latency Compensation

In non-frame accurate systems, Viz Mosart only compensates for delays in the behavior of the playout *video server*.

With frame accuracy, Viz Mosart can compensate for any latency of the controlled equipment.



As a pay-off of Predictable Behavior of Equipment, a calibration process can reveal the latency of each piece of gallery equipment.

Viz Mosart can then compensate for these delays, by always executing commands earlier.

In the figure above, the three devices (*video switcher*, *audio mixer* and *video server*) have millisecond latencies of 1, 2 and 3 frames respectively.

After compensating by executing each command specifically earlier, Viz Mosart can now successfully perform the TAKE command, synchronized for all three devices, from the same frame.

11.2 Requirements

- A clock source on the Viz Mosart server machines (main and backup). Either of the following options provides it:
  - **A Plura timecode reader card**, installed on the main and backup Viz Mosart server machines. This implies that the machine must have a free slot for this type of card.

The card takes digital genlock (SDI), and delivers the traditional digital VITC and ATC-VITC as timecode formats. For the models in use and the firmware each one needs, see Plura card models and firmware.

- **PTP**, from Viz Mosart 5.15.0 onward.
  Windows disciplines the system clock to a PTP grandmaster, and no additional card is needed. See PTP Setup.

- Calibration has been performed.
- A dedicated template set.
  There will be timing differences for template execution under frame accurate operation. Latencies are automatically compensated for primary events.

> **Info**: Whilst frame accurate operations are being continually developed and improved, not every feature from standard mode may be available or possible in frame accurate mode. This is also why a *dedicated template set* is essential.

11.2.1 Choosing a clock source

Both clock sources give the same frame accurate behavior. Pick the one that matches the facility:

Clock source	Viz Mosart version	What it needs
Plura timecode reader card	Any supported version	A card in each Viz Mosart server machine, with a genlock feed and a timecode feed wired to it.
PTP	5.15.0 or later	A PTP grandmaster on the network. No extra hardware in the server machine.

On versions before 5.15.0, and in any facility without a PTP grandmaster, a Plura card is the only route to frame accuracy. To use PTP instead, follow PTP Setup. Whichever clock source you choose, Viz Mosart still needs the latency of each controlled device, so run the tests in Device Calibration.

11.2.2 Plura card models and firmware

The card model and its firmware decide whether the video field interrupts arrive in the form Viz Mosart expects:

Card	Minimum firmware	Status
PCL PCIe 3G	3.16	Verified. The card generation Viz Mosart has always been used with.

Card	Minimum firmware	Status
PCL PCIe LV (v2)	9.25	In field use. Firmware 9.22 has an interrupt fault that Plura fixed in 9.25.
PCL PCIe 3G (v2)	9.27	In field use. Firmware 9.22 mislabels the odd and even video field interrupts, so Viz Mosart never accepts the external clock. Plura fixed this in 9.27, tested at 1080i50.

11.3 Enabling Frame Accuracy Operation

For frame accurate behavior, both Viz Mosart and the equipment it controls, require setup of both a genlock and timecode feed.

- **Genlock Settings**
  The switcher, video server (playout and recording) and Viz Mosart Server, all require genlock.
- **Timecode Feed**
  Both the recorder and Viz Mosart Server require a timecode feed.

Info: Story Recorder can be used without having proper genlock setup and a timecode reader card installed on the Mosart Server machine, *but only for testing*. In this case, the time of the day will be used.

11.3.1 Configuring Viz Mosart for Frame Accurate Operation

Once the system, including all equipment has been calibrated, frame accurate operation can be easily switched on or off.

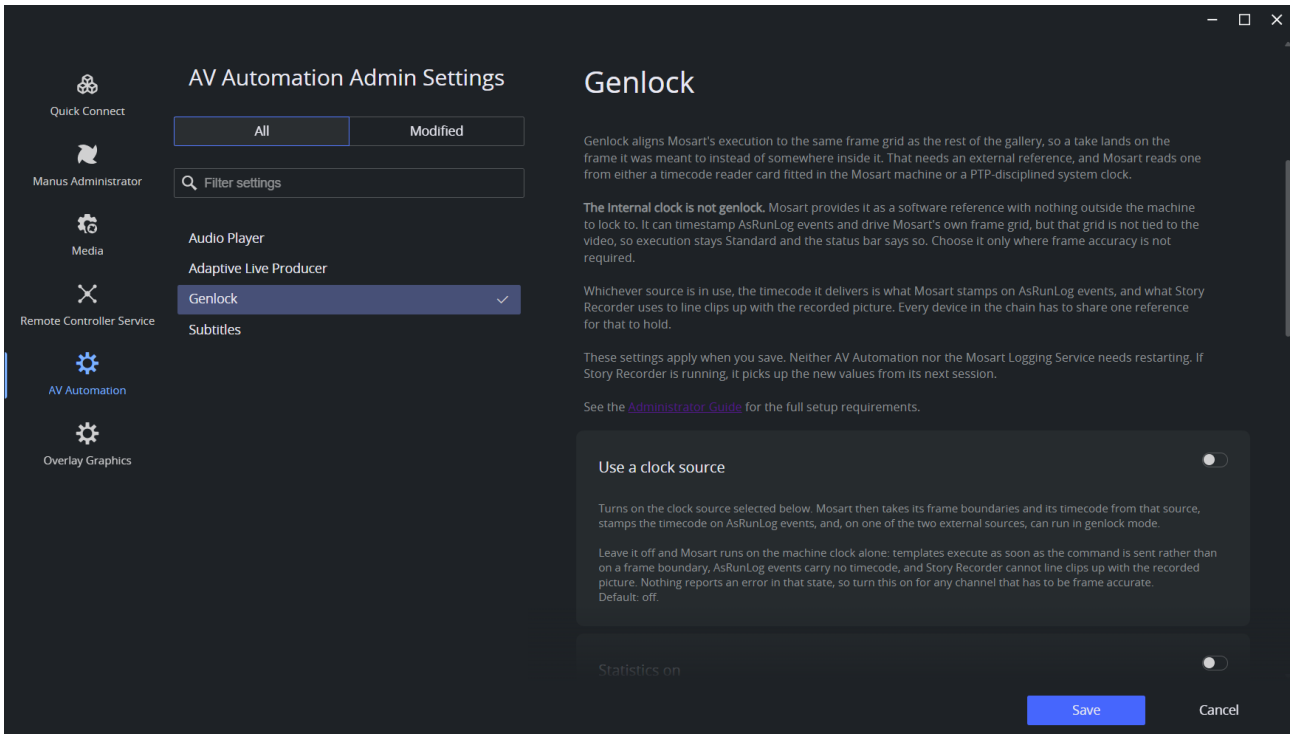
To activate frame accurate operations

Frame accuracy settings are made in **Viz Mosart Central Settings**.

1. In AV Automation, open **Central Settings**, then select **AV Automation > Genlock**. The **Genlock** tab under **Devices > Properties** links to the same place.
2. Set
   a. **Use a clock source**: switch on, then set **Clock source** to the clock the machine uses. Plura PCIe is the default. Select PTP if the Windows system clock is disciplined to a PTP grandmaster (Viz Mosart 5.15.0 and later).
   b. **Genlock mode**: switch on. Genlock needs an external reference, so this is available on Plura PCIe and PTP only, and is hidden while the clock source is Internal.
   c. **Calibration**: the individual latencies of the gear being controlled, as derived when calibrating the system. Each accepts a value from 0 to 100.
3. Click **Save**.
   The changes apply immediately if Story Recorder mode is not enabled. There is no need to restart AV

Automation or the Mosart Logging Service. If Story Recorder is enabled, the settings are taken into consideration in the next Story Recorder session.

Central Settings can be opened from Manus Administrator and Media Administrator as well, and a save there reaches AV Automation the same way.



Combinations of **Genlock mode** and **Clock source**:

Genlock mode	Clock source	Logic
Off	Not in use	Classic mode. No timecode in the Viz Mosart logs. No statistics can be performed.
Off	Plura PCIe or PTP	Classic mode. Timecode from the selected clock source is written in the Viz Mosart logs generated from AV Automation.

On	Plura PCIe or PTP	Frame accurate behavior when executing Viz Mosart templates during a show. Timecode from the selected clock source is written in the Viz Mosart logs, generated from AV Automation. Viz Mosart falls back to the *internal* clock source if the selected clock source is detected as inaccurate.
On	Internal	Not possible. The Internal clock is a software reference with nothing outside the machine to lock to, so it cannot drive genlock. Viz Mosart stays in Standard execution and the status bar reports Standard. **Genlock mode** is hidden while this clock source is selected, and switched off on save.
On	Not in use	Not allowed. Viz Mosart will fall back to Standard execution mode, with no timecode.

To deactivate frame accurate operations

1. In AV Automation, open **Central Settings**, then select **AV Automation > Genlock**.
2. **Genlock mode**: switch off.
3. **Use a clock source**: switch off.
4. Click **Save**.

11.3.2 Genlock and Timecode Reliability

Viz Mosart contains a high resolution internal genlock clock as a built-in safety mechanism which takes care of intermittent or lost external genlock and/or timecode signal.

- Whenever the external *genlock signal* drops out or becomes unreliable, Mosart will fallback to the internal genlock clock. Mosart will return to external once the signal is reliable again.
- Mosart effectively runs on the *internal timecode clock*, constantly synchronized to the external clock when available, keeping Viz Mosart in phase.

11.4 UTC or Local Time

Timecodes carry a time of day, and a time of day only has meaning together with a convention: UTC or local time. Two settings tell Viz Mosart which convention is in use:

Setting	Location	Meaning
External clock is local time	Central Settings > AV Automation > Genlock	Describes the signal: whether the timecode from the master clock carries local time or UTC.

Setting	Location	Meaning
Use local time	Central Settings > Manus Administrator > General	Chooses the convention Viz Mosart itself uses for timecodes, including every Story Recorder EDL and render command.

When the two differ, Viz Mosart converts the incoming timecode to its own convention by adding or subtracting the UTC offset, so Viz Mosart's own clock works with any combination.

The video server sets the rule. Story Recorder computes every mark as the event timecode minus the Start of Media that the video server reports for the recording. That Start of Media is used exactly as delivered; Viz Mosart never converts it. Both values must be in the same convention, which reduces to one rule: **Use local time** must match the time reference of the master clock that feeds the video server. The rule covers every Story Recorder output: the Vantage and Mosart EDL files, and the render and export commands sent to Mimir and TriCaster.

Master clock output	Use local time	Story Recorder result
UTC	`false`	Correct marks
Local time	`true`	Correct marks
UTC	`true`	Every mark offset by the UTC difference
Local time	`false`	Every mark offset by the UTC difference

How a mismatch shows up. Every mark is wrong by the same amount, and that amount equals the difference between local time and UTC (for example, 5 hours when the local time zone is US Eastern). When Story Recorder is enabled, Viz Mosart also logs a warning that the video server time differs from the Mosart time. Treat this as a configuration error; it cannot be calibrated away.

Note: Local time jumps with daylight saving; UTC does not. Facilities that record across a daylight saving change often run the house clock on UTC.

Info: A third setting also named `UseLocalTime` exists in `LogRepositoryConfig.xml`. It affects only the timestamps written to the log files, never the timecodes or the marks. Leave it at `false` so support can correlate the logs with other systems.

11.5 Calibrating the System

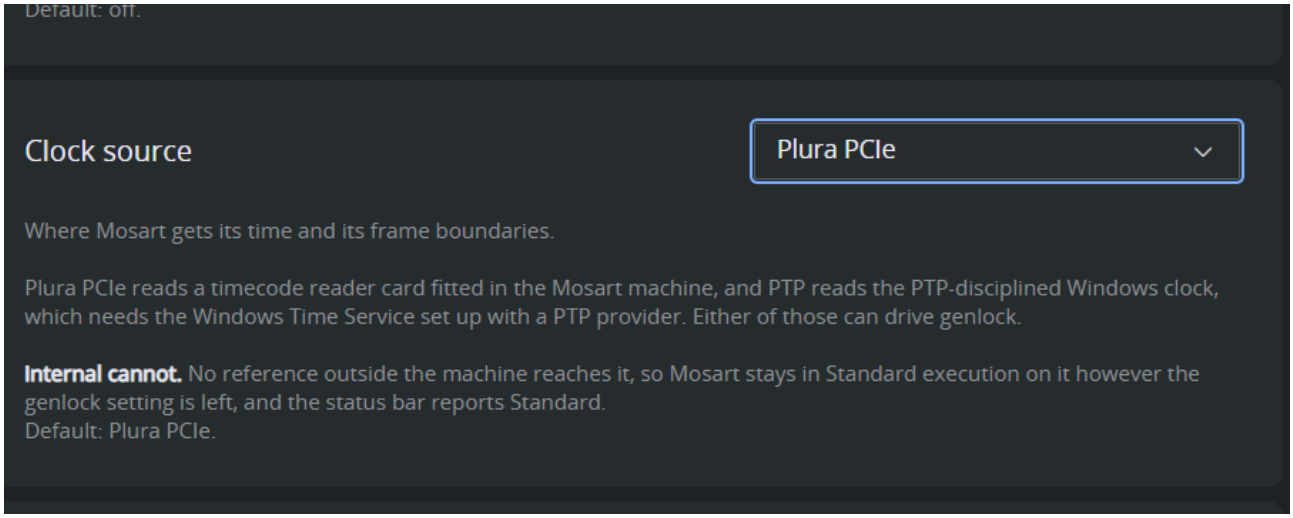
- For Viz Mosart to function in frame accurate mode, a calibration process has been created to better understand the timings associated with the gear being operated.
  Contact Vizrt Support for further details.

- The guide defines *six tests that must be executed* to detect the *latencies* (delays) introduced when operating gallery gear, so that Viz Mosart can compensate for these delays and verify equipment reliability.

The tests, the values they produce, and where to enter them are described in Device Calibration.

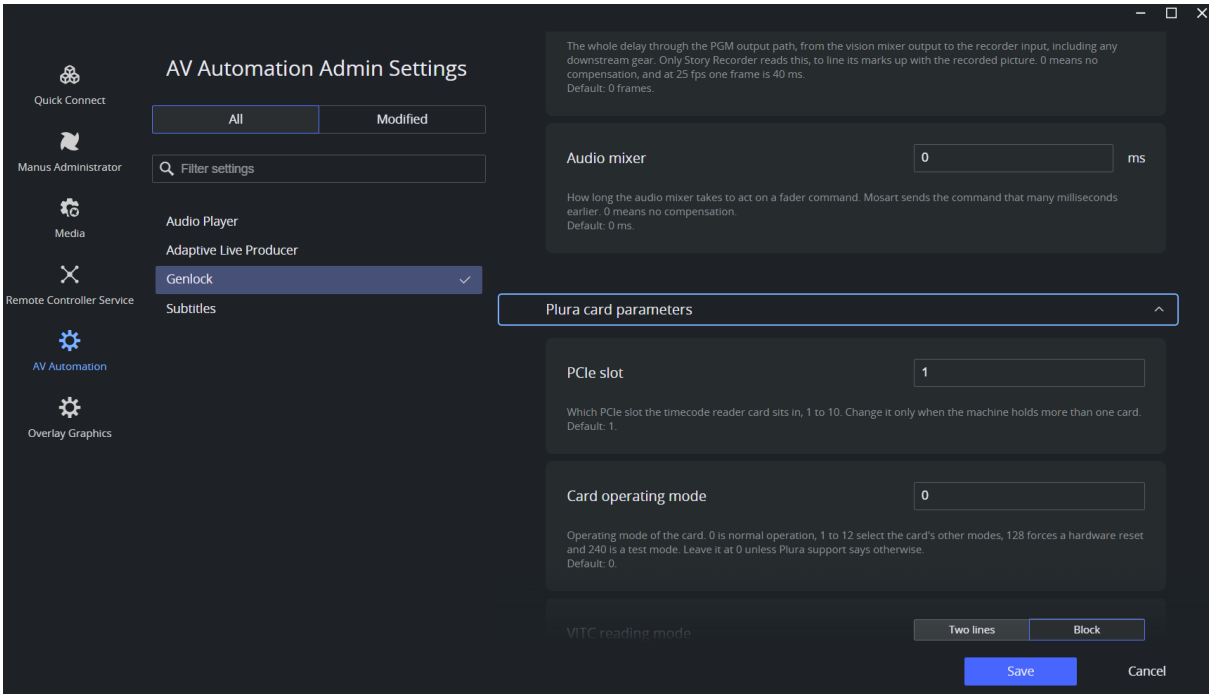
11.5.1 Timecode Readers

The properties of the Plura timecode reader card are set in **Viz Mosart Central Settings**, under **AV Automation > Genlock**, in the **Plura card parameters** section. The section appears only while the clock source is Plura PCIe.



To configure the Plura timecode reader card

1. In AV Automation, open **Central Settings**, then select **AV Automation > Genlock**.
2. Expand **Plura card parameters** and set the PCIe slot, the card operating mode, the VITC reading mode and line range, the reader priority, the timecode frame rate, and whether mid-frame events are raised. The defaults suit a standard installation; change them only when the card is fitted or fed differently.
3. Click **Save**. The card is reconfigured without restarting AV Automation.



11.5.2 Gathering Clock Source Statistics for Vizrt Support

Once Viz Mosart is set up for frame accuracy, it is recommended to perform clock source statistics that will verify the reliability of the genlock and timecode signals.

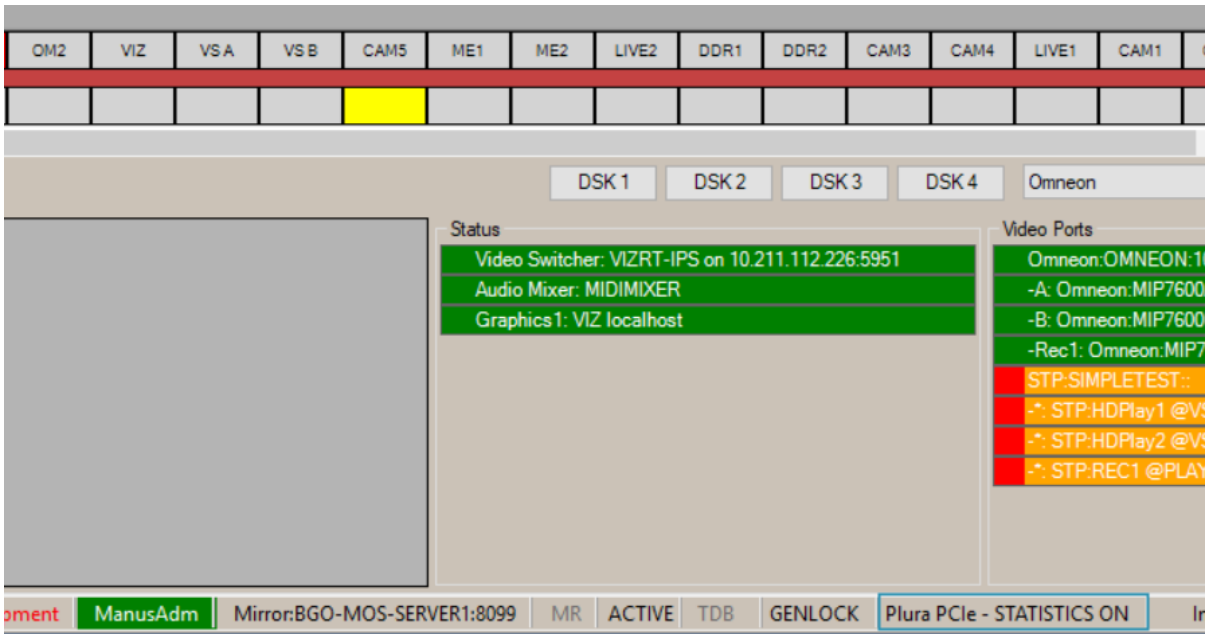
Follow the procedure below, and forward the results to Vizrt customer support, who will perform an analysis and report results.

To start clock source statistic gathering

1. In AV Automation, open **Central Settings**, then select **AV Automation > Genlock** and set:
   a. **Use a clock source** on, with a **Clock source** selected.
   b. **Genlock mode** on.
   c. **Statistics on**: switch on.
2. Specify the location where you wish to save the stats in **Statistics log path**.
3. Click **Save**.
4. Check the indicator in the AV Automation status bar at the bottom of the screen, that statistics are on:
5. Collection stops on its own once 10,000 samples have been gathered. Viz Mosart takes one sample per frame, so how long that takes depends on the frame rate: about 6 to 7 minutes at 25 fps, about 5 and a half minutes at 29.97 fps, and roughly half those times at 50 and 59.94 fps.
   You can stop the statistic measurements earlier by switching **Statistics on** off, as described in step (**1**) above.
6. Collect the saved statistics from the location defined in **Statistics log path** (above). Viz Mosart writes three files. Send all three to Vizrt Support:
   a. `GenLockStats_FrameClock_<uid>.csv` : the statistics for the Mosart frame clock, driven by the selected clock source (internal or external).

b. `GenLockStats_ExternalCS_<uid>.csv` : the statistics for the configured external clock source.
c. `GenLockStats_Summary_<uid>.txt` : a readable summary that carries the conclusions. Support finds this one the most useful of the three. One line is worth checking yourself: `IsTcSyncedWithExternal = True` confirms that the timecodes are synced with the external source. `False` means Viz Mosart is frame-locked but the timecodes come from the PC clock, and every Story Recorder mark will carry a constant offset.

The `<uid>` part is a date and time stamp in the form `yyyyMMdd-HHmmss` , so the highest value is the most recent run.

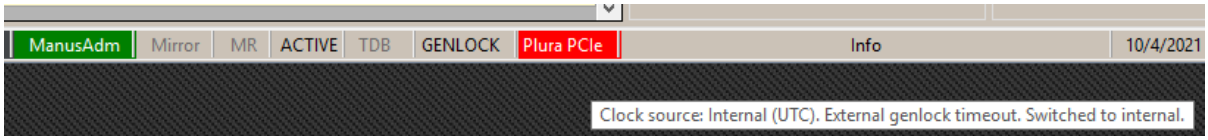


Tip: You can start/stop the clock source statistics by toggling the keyboard shortcut **Ctrl+Shift+I** directly in AV Automation, without opening Central Settings.

To analyze clock source statistics

This is a complex operation performed by Vizrt Support.

- The frame clock (FrameClock) is used internally by Viz Mosart to execute templates frame accurately.
- If the external clock source is not reliable (shown in red on the AV Automation status bar), the frame clock is driven by the internal clock source:
- If the external clock source is reliable, the internal FrameClock should behave more or less as the external clock source.
  This can be seen in MS Excel by comparing the chart generated (in Excel) from GenlockPeriodMs or JitterMs columns in `GenLockStats_FrameClock_<uid>.csv` file,
  with the chart generated from the same columns in `GenLockStats_ExternalCS_<uid>.csv` .



11.6 Genlock Logging

A new type of logging has been introduced with frame accuracy in Viz Mosart to help identifying problems related to genlock, if this is enabled.

- To enable the genlock logging, copy the configuration file **GenLogRepositoryConfig.xml** from the *Viz Mosart installation folder/Mosart Server/ConfigurationFiles* to your working folder (*Channeltemplates* or *ProgramData*).
- Open **GenLogRepositoryConfig.xml** in an editor and configure it in a similar way as the other type of logging (refer to System Logging). These configurations also include settings about when the logging files shall be purged.
- The genlock logging is by default generated in *C:\MMLogs\Genlock* every time a new show is run in Mosart (whether Story Recorder mode is enabled or not).
- A new genlock logging file is generated when a rundown is (re)loaded in Viz Mosart or when the show is paused whilst Story Recorder is enabled.

Tip: Contact Vizrt Mosart Support for assistance with
- Statistics analysis
- Calibration

11.7 Frame Accuracy in Story Recorder

For Story Recorder to provide high quality final shows, frame accuracy is a requirement. SR can also be run with Viz Mosart running in Standard mode, *but only for testing*.

See also the section **Story Recorder** of the *Viz Mosart User Guide*, and Show Design for Frame Accuracy and Story Recorder.

11.8 Limitations

Frame accuracy is only available on events driven from primary templates (clips, switcher, audio and full screen graphics), not secondary events (like overlay graphics or accessories).

11.9 Device Calibration

- Device Calibration
- General setup
- Calibration Overview
- Before you Start
- Test Clips
- Rundown Recipes
- A-series Calibration Tests: The Recommended Set
- B-series calibration tests: alternative tests

11.9.1 Device Calibration

For Viz Mosart to operate frame accurately, the system must be calibrated so that Viz Mosart knows the latency of each piece of gallery equipment it controls. This section describes the calibration tests that reveal those latencies, and where to enter the measured values.

Introduction

For correct frame accurate performance, Viz Mosart depends on several things:

1. A proper genlock setup (described below).
2. Gear supporting frame accurate performance.
3. A proper calibration session during setup.

Viz Mosart currently supports frame accurate and deterministic control of the switcher, the audio mixer, and the video server, provided the gear involved supports this. Frame accurate control will be expanded to more devices in future updates.

About deterministic behavior and latency

For frame accurate control to work, the gear being controlled must support it. This comes down to two main traits.

First, the gear must have a genlock mode, accept a genlock feed, and respond to incoming commands aligned to the start of one of the following frames. One exception is audio gear, which often does not support genlock. Audio gear should still respond predictably to incoming commands.

Second, gear may have some latency (delay) in its response to vital incoming commands such as cross point switching, audio fader changes, and clip start. That delay must always be the same, because only then can Viz Mosart compensate for it.

For example, a video server may take 7 frames to respond to a clip play command, but it must always take 7 frames for that command.

What frame accurate control means

Frame accurate control means two things:

- You get deterministic (repeatable) behavior from the gear being controlled.
- Viz Mosart compensates for any latency (delay) involved when controlling that gear.

Copyright © 2026 Vizrt

To make this happen, all the gear involved must be connected to the same genlock network, and the Viz Mosart server must run from the same timing reference. Viz Mosart must also learn how the controlled gear behaves in terms of latencies.

For example, to start a clip frame accurately, Viz Mosart must know two things:

- Exactly how many frames it takes the video server to respond to a Play command.
- Exactly how many frames it takes the switcher to respond to a cross point command.

The calibration process finds these delay values and configures Viz Mosart accordingly. The following sections present the calibration process as a series of tests.

What frame accurate control does not mean

- Frame accurate control does not mean that everything suddenly happens in an instant, without any delays. Viz Mosart cannot speed up the gear it controls.
- Frame accurate control does not mean that Viz Mosart can squeeze predictable, deterministic behavior out of unpredictable gear.
- Frame accurate control does not mean that Viz Mosart can produce reliable results controlling gear that has not been properly set up with genlock and timecode.
- Frame accurate control does not mean that Viz Mosart can produce reliable results out of a machine room that has not been properly set up and calibrated.

11.9.2 General setup

To get frame accurate behavior, both Viz Mosart and the gear it controls require several setup steps, described here.

Note: Most of these steps are obvious and have probably already been applied. They are mentioned for completeness.

Genlock

The studio equipment has to behave as frame accurate gear in its own right, so at least the following must be connected to the same genlock network:

- The video switcher, for frame accurate cross point switching.
- The video server, for frame accurate file playback.
- The video recorder, if any, for frame accurate recording.

Other gear in the setup may also need genlock, but these are the main players. The format in which genlock is delivered does not matter (analog blackburst or SDI), and mixing these formats is fine, provided the overall timing is tight between the connected devices.

The Viz Mosart server needs a reference of its own, and where that reference comes from depends on the clock source you run. With a Plura timecode card, the server takes a genlock feed into the card, so it joins the same genlock network as the gear listed above. From Viz Mosart 5.15.0 you can run PTP instead. Windows then disciplines the system clock to a PTP grandmaster over the network, and the server needs no genlock feed and no card.

PTP changes only how the Viz Mosart server itself gets its reference. The video switcher, the video server and the video recorder still need genlock on both routes. See PTP Setup.

Timecode

In addition to genlock, the video recorder, if any, must be connected to the timecode house clock network. Different timecode formats can be mixed, provided the connected devices receive the same timecode for the same frame.

Whether the Viz Mosart server needs that same house timecode depends on the clock source. On the Plura route it does, and the Plura card delivers the timecode to the server. Under PTP it does not, because Windows disciplines the system clock to the PTP grandmaster instead.

Viz Mosart Server

A Plura timecode card (such as the PCIe 3G) connects the Viz Mosart server to both the genlock and the timecode network. The same Plura card delivers timecode to the Viz Mosart server, and both digital VITC and ATC-VITC are supported.

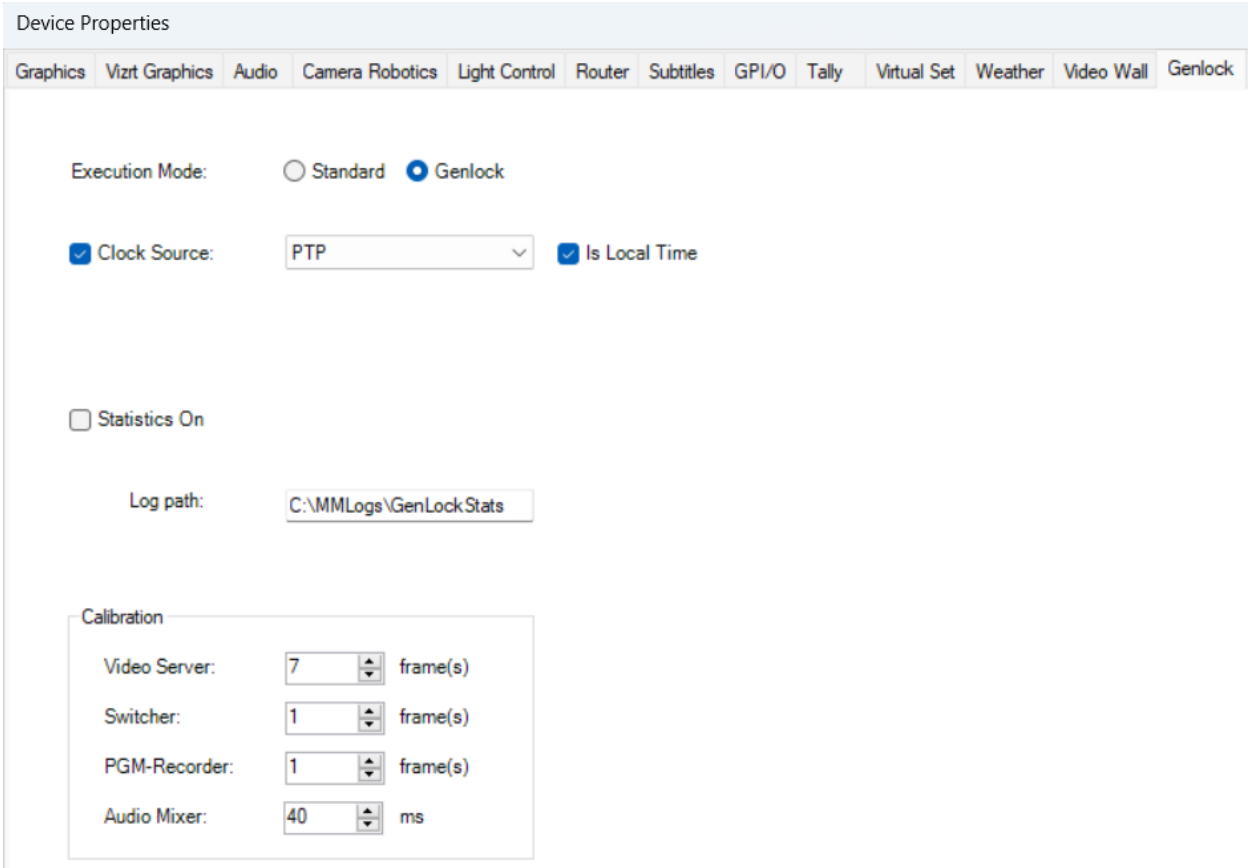
From Viz Mosart 5.15.0, the clock source can be PTP instead. Windows disciplines the system clock to a PTP grandmaster, so no card has to be installed in the server computer. See PTP Setup.

Viz Mosart Genlock Settings

In AV Automation, go to **Devices > Properties > Genlock** and set Viz Mosart in Genlock mode. Then set the clock source to match the route you run, either the Plura card or PTP.

Enter the latencies measured by the tests described here on the AV Automation **Genlock** tab, in the **Calibration** box.

Copyright © 2026 Vizrt



The four fields in the **Calibration** box are:

Field	Measured by	Unit
Video server command latency	Test A2 or B2	Frames
Switcher command latency	Test A3	Frames
PGM-Recorder latency	Test A4	Frames
Audio mixer command latency	Test A6	Milliseconds

Video Recorder Settings

For Story Recorder users only, set the video recorder to an **I-frame only** format for the intermediate recordings. This works around some issues with long-GOP format recordings. It has no impact at all on the final, consolidated show clips.

Video Server Settings

All the video server ports used for clip playout through Viz Mosart must have identical play command latency values. That is, these ports must behave in identical ways. One of the calibration tests described below focuses on this.

Info: Viz Mosart cannot perform frame accurately with ports that have different latency values.

Audio and video sync

The audio-video lip sync on the final PGM output is expected to be perfectly set up. One of the calibration tests below involves audio mixer latency measurements, and that test only makes sense if the system is correctly set up to begin with.

11.9.3 Calibration Overview

The table below lists the available calibration tests. There are two series: the recommended A-series, and the alternative B-series tests that can be considered where available.

When setting up frame accurate mode for the first time, perform all tests in order, choosing the A or B series where available.

Once set up, the calibration tests only need to be performed again after changes in the machine room setup, or firmware updates of the gear being controlled.

Test	Alternative	Needs first	Gear	Latency value	Optional	Default	Description
A1		-	Timecode burn-in feed, recorder, Clip Tool	Recorder in-out latency	Optional	0 frames	Any latency added by the video recorder.
A2	B2	A1 (if you ran it)	Recorder, Clip Tool, re-cabling	Video server command latency		1 frame	The time it takes the video server to respond to a play command and start playing.

Test	Alternative	Needs first	Gear	Latency value	Optional	Default	Description
A3		A2 (not B2)	Mobile phone or recorder	Switcher command latency		1 frame	The time it takes the video switcher to respond to a cross point switch command.
A4		A3	Two camera feeds, Story Recorder, Clip Tool	PGM-Recorder latency		0 frames	The sum of in-out latencies introduced by the video switcher, any downstream gear in the PGM output path, and the video recorder. Relevant for Story Recorder only.
A5		-	Recorder, NLE	Audio and video lip sync	Optional	-	Test of audio-video lip sync offset as set up in the server room.
A6		A2 and A3	Recorder or Story Recorder, Clip Tool or NLE	Audio mixer command latency		0 ms	The time it takes the audio mixer to respond to a fader command.
B2		- (assumes switcher = 1 frame)	Mobile phone	Video server command latency		0 frames	

11.9.4 Before you Start

Gear for the whole session

- A video recorder able to record a single port and the PGM output.

- Harmonic Clip Tool, or an equivalent that shows per-frame timecode.
- A professional NLE (Premiere Pro, Final Cut Pro) for the two audio tests.
- A mobile phone that can film a monitor, if you run A3 or B2.
- Two camera feeds with visibly different pictures, for A4.
- Cables to patch a playout port directly into the recorder input, for A2.
- The test clips, at your own frame rate. See **Test clips**.

The rule every test follows

Set the latency you are about to measure to 0, and make sure the latencies you already measured are entered on the **Genlock** tab. Test B2 is the one exception: it starts at 20 frames and you subtract what you find.

What feeds the Calibration box

A1 and A5 produce no setting of their own. A1 gives you a number you subtract inside A2. A5 is a sanity check on the machine room before Viz Mosart is involved at all.

11.9.5 Test Clips

The calibration clips are built for these tests and are supplied with the calibration package. They must match the frame rate and raster of the house standard, so a facility running anything other than 25 fps needs its own set. Build them to the specification below.

Clip	Video	Audio	Used by
`WhiteBlack-Clapper-Num`	Frame 0 is a single white frame carrying a large blue **0**. Every following frame is black with its frame number burned in, counting 1, 2, 3 and up.	A clapper transient at full level starting exactly on frame 0.	A2, A3, A5, B2
`Yellow-Silent-TC`	Solid yellow with burned-in timecode.	Silent.	A3, A6, B2
`Pink-Sine-TC`	Solid pink with burned-in timecode.	Sine tone at full level from the first frame, with no fade-in.	A6

Rules for a home-made set:

- Same frame rate and raster as the house standard.
- I-frame only, about 10 seconds long.
- The white frame in `WhiteBlack-Clapper-Num` is exactly one frame.

- No audio fade at the head of `Pink-Sine-TC`. A6 measures audio onset against the first pink frame, so an encoder-added ramp silently corrupts the result.

11.9.6 Rundown Recipes

These are the breakdowns of the test rundowns mentioned in the calibration tests. Test rundowns are not easy to make available, so create your own set as described below.

> **Info:** Use the indicated clips. They were specifically designed for the calibration process.

Rundown Calibration-A2

1. Camera
2. Clip `WhiteBlack-Clapper-Num` playout port A.

Rundown Calibration-A3

1. Camera
2. Clip `Yellow-Silent-TC`, auto-take 5s, A/B roll.
3. Clip `WhiteBlack-Clapper-Num`, A/B roll.

Rundown Calibration-A4

1. Camera 1
2. Camera 2, with a visually different feed than Camera 1.

Rundown Calibration-A5

1. Camera
2. Clip `WhiteBlack-Clapper-Num`, A/B roll.
3. Camera.

Rundown Calibration-A6

1. Camera
2. Clip `Yellow-Silent-TC`, A/B roll, fader with audio ripple, crossfade in/out set to 0, autotake 5s.
3. Clip `Pink-Sine-TC`, A/B roll, fader with audio ripple, crossfade in/out set to 0.

Rundown Calibration-B2

1. Camera
2. Clip `Yellow-Silent-TC`, A/B roll.
3. Clip `WhiteBlack-Clapper-Num`, A/B roll.

11.9.7 A-series Calibration Tests: The Recommended Set

Calibration A1: Recorder in-out Latency (optional)

This test is optional, but still recommended if you have the required gear available.

The video recorder is required in several calibration tests, and in some of these it is important to know whether the video recorder introduces any in-out (throughput) latency. For example, if the recorder is recording a clapper board hit at a given timecode, will it show up in the recording at the same timecode, or a few frames later?

A professional video recorder normally has zero in-out latency, and your recorder will likely behave the same. If in doubt, perform the test.

What is the plan?

Connect a feed with burned-in, actual house clock timecode directly to the recorder input and make a short recording. No Viz Mosart is involved here. Then check the recording and compare the difference between the burned-in timecode and the recorder's own timecode track. The difference is the recorder latency.

Setup

This test is somewhat demanding in terms of gear. You need a feed showing the live house clock timecode burned in. This can be achieved in several ways:

- A zero-latency camera pointed at a zero-latency timecode display.
- A zero-latency camera with built-in timecode display.
- A device such as a Rubidium, emitting an SDI feed with zero-latency burn-in timecode.

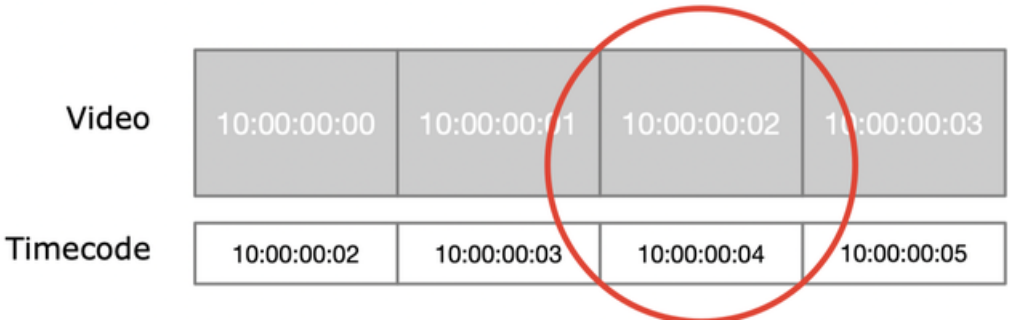
Some re-cabling is also needed for this test.

Test material and gear

- Timecode feed as described above.
- The video recorder being tested, properly set up with genlock and timecode.
- A tool to inspect the recorded video, such as Harmonic's Clip Tool.

Test

1. Connect the timecode feed directly to the recorder input.
2. Make a recording a few seconds long.
3. Inspect the recording using Clip Tool (or similar).
4. Establish the difference in frames between the burn-in timecode shown in the video area and the timecode associated with the frame (from the recorder's timecode track). In the figure below, the latency found is 2 frames.
5. Remember this latency as the **recorder in-out latency**. It is needed later.



In the example above, the 10:00:00:02 timecode from the feed being recorded is matched with a 10:00:00:04 timecode from the video recorder's own timecode track. The recorder in-out latency here is 2 frames.

Calibration A2: Video Server Command Latency

This test measures the video server command latency, which is the time it takes the video server to respond to a play command, expressed in frames. Perform the test for each of the individual video server ports involved in the playout of rundown clips.

What is the plan?

Cue a clip in the video server that starts with a single white frame, followed by black frames. At a known timecode, Viz Mosart instructs the video server to play the clip while the playback port is being recorded. Check the recording and look for the recorder timecode associated with the last white frame, which is the first frame being played. The difference in timecodes is the latency.

Setup

This setup needs some re-cabling. To measure the latency of playout port A, for example, use a cable to connect port A directly to the recorder input. Re-cable to test the other ports.

Test material and gear

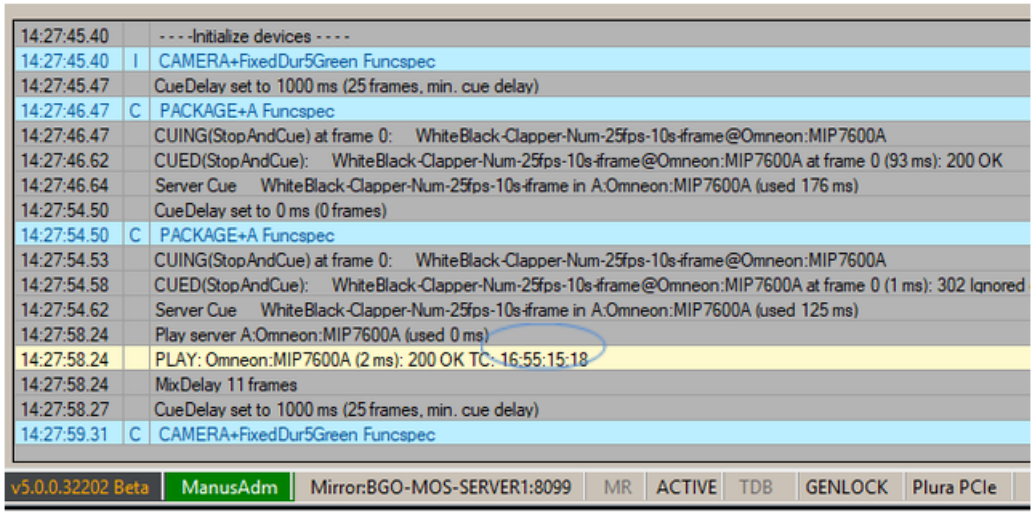
- Rundown: `Calibration-A2`. See Rundown recipes above.
- A video recorder. If you do not have a recorder, run alternative test B2 instead.
- A tool to inspect the recorded video, such as Harmonic's Clip Tool.

Test

1. Cable the port being tested directly to the recorder input.
2. On the AV Automation **Genlock** tab, set video server command latency to 0.
3. Load rundown `Calibration-A2`.
4. Start the recording.
5. Start the rundown and run story by story.
6. A few seconds into the clip, stop the recording.
7. In the AV Automation Log area (see Figure 3 below), look up the timecode for the moment the video was started. Call this TC1 (the blue circle in Figure 2 below).
8. In the recording, using Clip Tool or similar, look up the timecode of the last white frame. This is TC2 (the red circle).
9. The command latency for this port is the difference in frames between TC1 and TC2. In the figure below, that is 3 frames of latency.
10. If you measured the recorder in-out latency, subtract that value from the latency found in the previous step.
11. On the AV Automation **Genlock** tab, set video server command latency to the latency found.

Repeat this test several times for the same port, to check for consistent behavior. Then repeat the test for each of the playout ports.

> **Info:** All playout ports must have identical latency values.



Calibration A3: Switcher Command Latency

This test only makes sense, and only produces a reliable result, if you were able to run the recommended test A2 beforehand rather than the less preferred test B2. If you had to use test B2, which assumes a switcher command latency of 1 frame, skip test A3 and keep the 1 frame value.

This test measures the switcher command latency: the time it takes the switcher to respond to a cross point command, expressed in frames.

What is the plan?

While a yellow clip is playing, cue and start a second clip with burned-in increasing frame numbers (0, 1, 2 and so on) that starts playing with frame #0 exactly at the take point. Video server latency was measured in the previous test. At the same time, open the cross point for the clip and record the PGM output. If frame #2 shows as the first frame of the clip instead of the expected #0, the switcher has a 2-frame command latency.

Test material and gear

- Rundown: `Calibration-A3`. See Rundown recipes above.
- A mobile phone or video recorder for recording the PGM output, for example using AV Automation or Harmonic's Clip Tool to start the recording on the video server.

Test

1. Make sure cabling is restored to normal.
2. On the AV Automation **Genlock** tab, set switcher command latency to 0.
3. On the AV Automation **Genlock** tab, make sure that video server command latency is calibrated and specified.
4. Load rundown `Calibration-A3`.
5. With a mobile phone or recorder, record a video of a monitor showing the PGM output.
6. Start the rundown.
7. Stop the recording once the rundown is completed.
8. In the recorded video, locate the first non-yellow frame. The big blue number shown is the latency. See Figure 4 below as an example.
9. On the AV Automation **Genlock** tab, set the switcher command latency value to the number found.
10. To confirm, repeat the test from step 4.
11. The first non-yellow frame should now be a white frame with a big blue 0, as shown in Figure 4B below. Success.

Repeat this test at least 5 times. All tests should result in the same latency value.

Fig. **4A**: Switcher command latency is 1 frames.

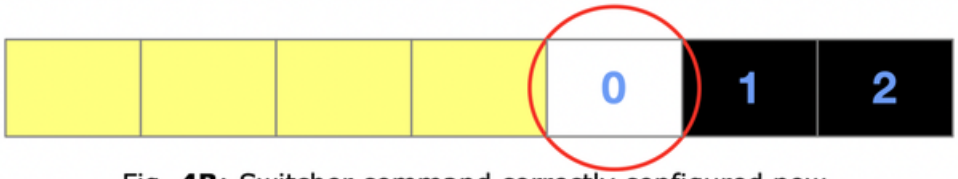
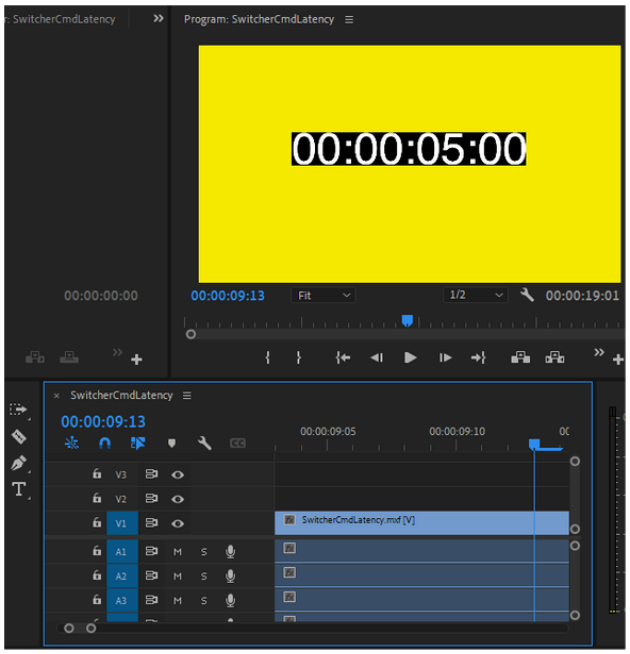
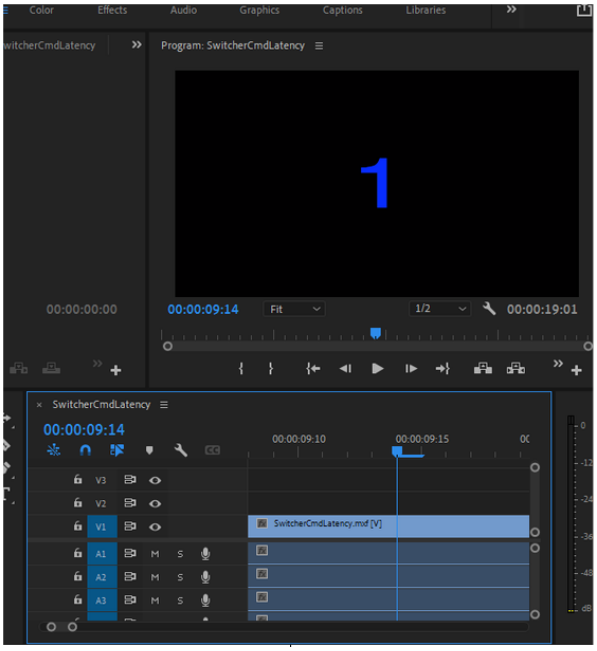


Fig. **4B**: Switcher command correctly configured now.





Calibration A4: PGM-recorder Latency

> **Info:** This test is relevant for Story Recorder users only.

This test measures PGM-recorder latency, which is the sum of in-out latencies from the vision mixer, any additional gear in the PGM output path, and the in-out latency of the video recorder used for Story Recorder.

What is the plan?

In Story Recorder, take Cam1 followed by Cam2 while recording the PGM output. Use the Story Recorder panel to get the house clock timecode for when Cam2 was taken. Then check the recording, locate the cut to Cam2, and find the recorder timecode for that moment. The difference between the two timecodes is the PGM-recorder latency.

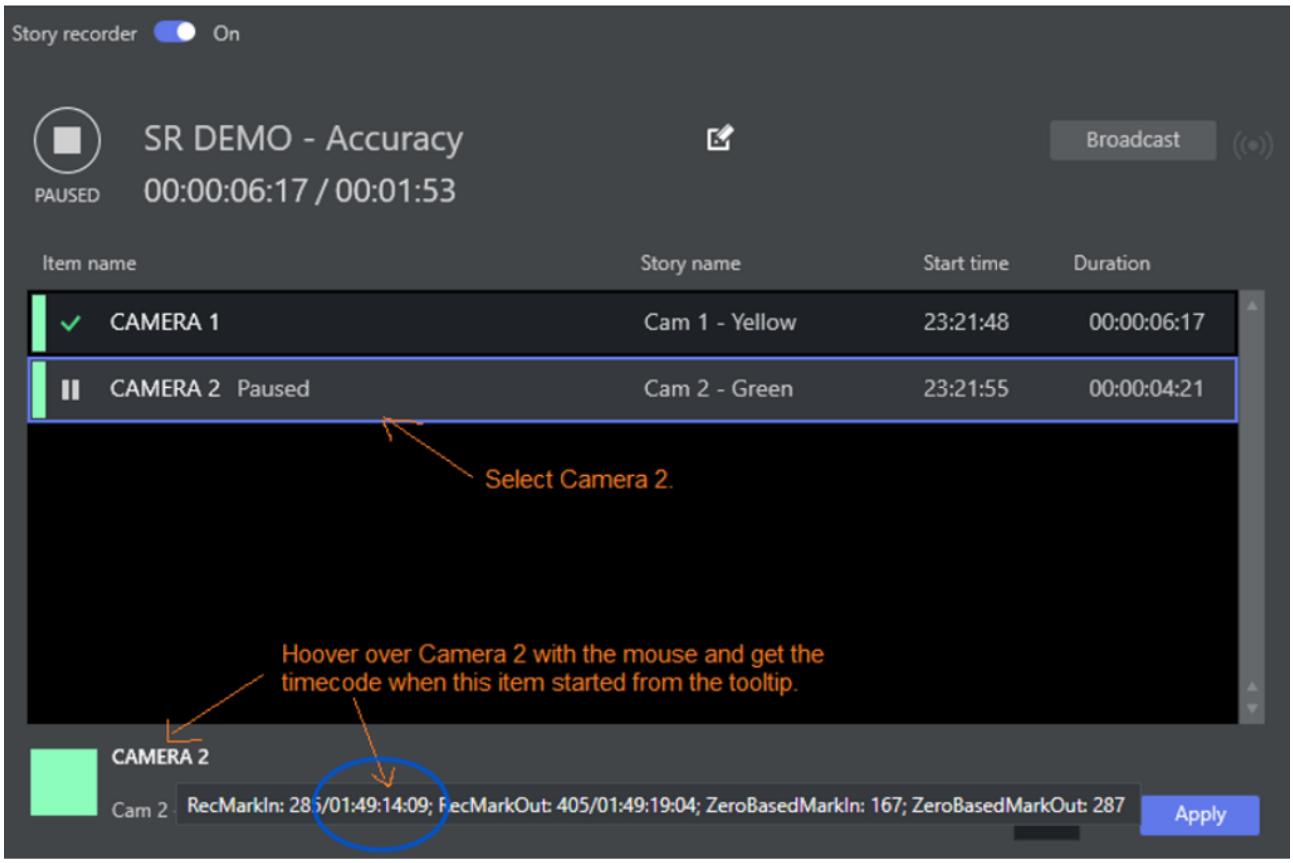
Test material and gear

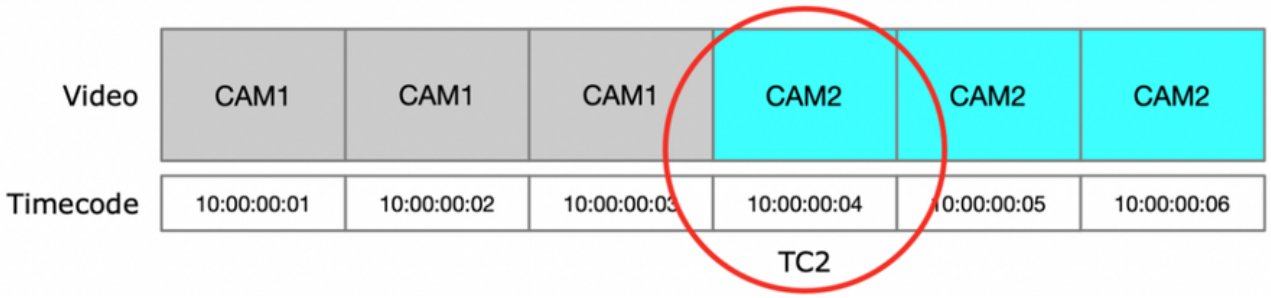
- Rundown: `Calibration-A4`. See Rundown recipes above.
- Two camera feeds.
- Story Recorder's video recording feature.
- A tool to inspect the recorded video, such as Harmonic's Clip Tool.

Test

1. On the AV Automation **Genlock** tab, make sure the switcher command latency value is calibrated and specified.
2. Load rundown `Calibration-A4`.
3. Enable Story Recorder. Wait for it to enter Record mode.

4. Start the rundown.
5. Once the rundown has ended, pause Story Recorder.
6. In the Story Recorder panel on screen (see Figure 7 below), locate the timecode of the take of Camera 2. Call this TC1 (the blue circle).
7. Using Clip Tool or similar, load the recorded clip and locate the hard cut from cam1 to cam2.
8. Note the recorder timecode of the first frame of the Camera 2 take. Call this TC2 (the red circle).
9. On the AV Automation **Genlock** tab, set PGM-recorder latency to the number of frames between TC1 and TC2.





Calibration A5: Audio and video lip sync (optional)

> **Info:** This test is optional and is defined here for completeness, as this has probably already been taken care of in your setup.

This test checks the lip sync relation between video and audio as found on the PGM output. The goal is to have audio and video perfectly in sync, even before Viz Mosart comes into play.

What is the plan?

Play a clapper-board style clip into an open cross point with open audio faders, record the PGM output in a video recorder, and check the result. This does not test Viz Mosart, only the PGM signal.

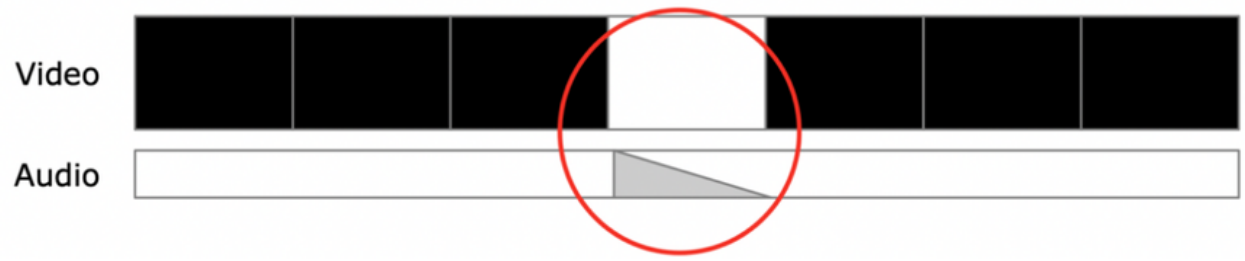
Test material and gear

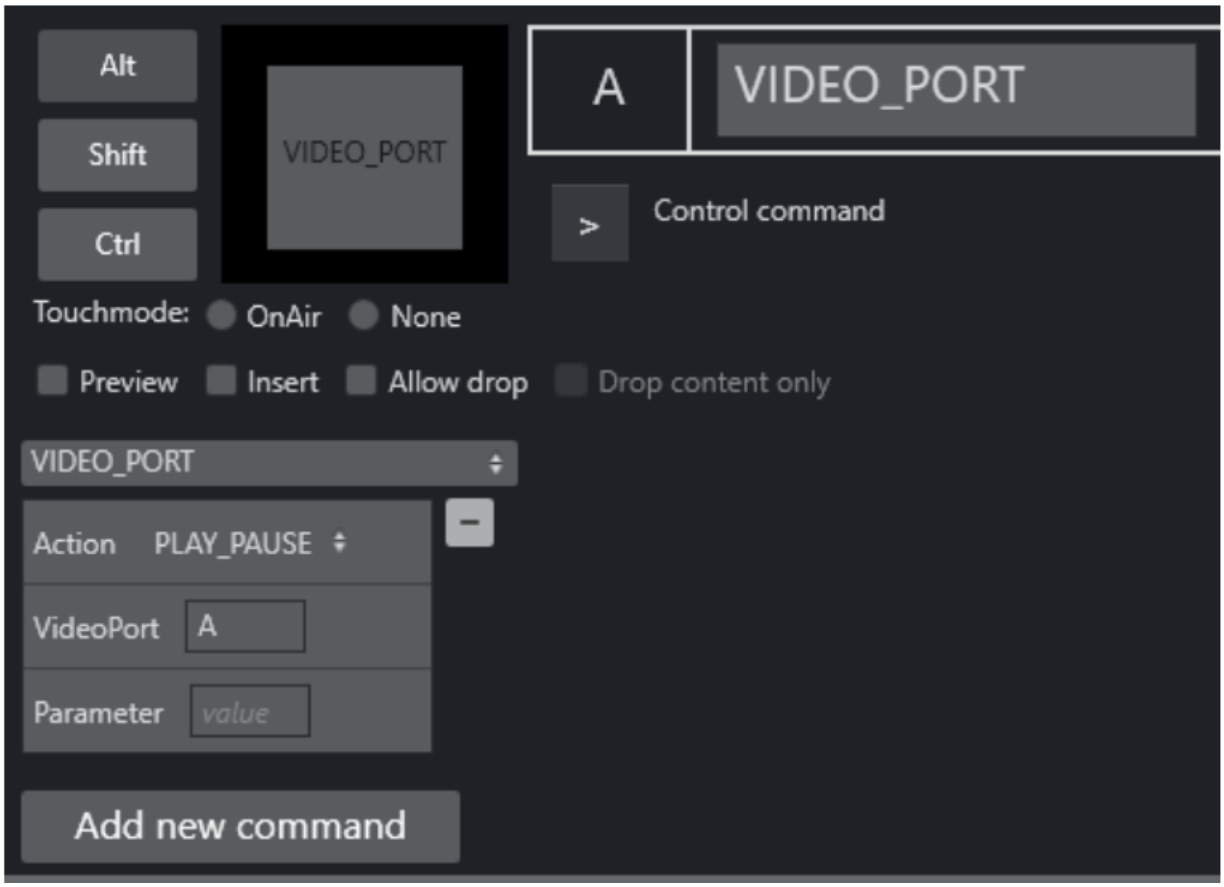
- Rundown: `Calibration-A5`. See Rundown recipes above.
- A video recorder for recording PGM.
- A professional NLE such as Premiere Pro or Final Cut Pro.

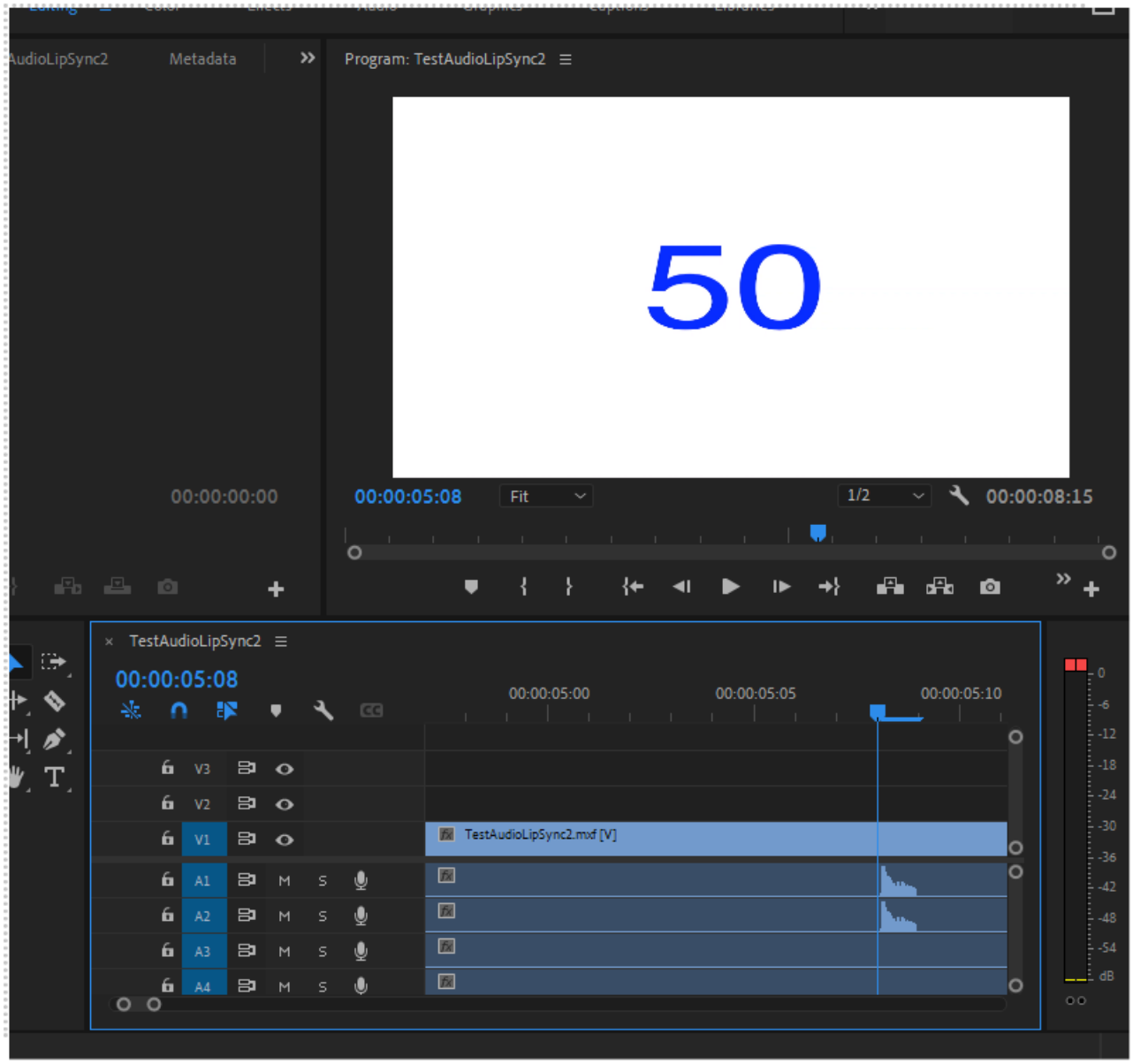
Test

1. Load rundown `Calibration-A5` and Set as Next the clapper clip, if this is not the first story in the rundown. This cues the clapper clip on PGM.
2. Observe which port is cued on PGM.
3. Prepare a keyboard shortcut (see Figure 10 below) instructing the video server to play the clip on the port cued on PGM.
4. Open the audio fader for the same port in the AV Automation UI.
5. Start the recorder.
6. Press the keyboard shortcut to start the clip.
7. Stop the recording after about 10 seconds.
8. Using an NLE (or other reliable software), check whether video and audio are in sync. The clapperboard sound should start within the white frame, close to the start of the white frame. See the figure below.

If audio and video are not in sync, tweak the server room gear accordingly and repeat the test until satisfied.




Alt
Shift
Ctrl
VIDEO_PORT
A
VIDEO_PORT
>
Control command
Touchmode:
OnAir
None
Preview
Insert
Allow drop
Drop content only
VIDEO_PORT
Action
PLAY_PAUSE
VideoPort
A
Parameter
value
Add new command



Calibration A6: Audio mixer command latency

The goal of this test is to measure audio mixer command latency: the time it takes the audio mixer to respond to a fader command.

What is the plan?

Take a silent yellow clip, followed by a pink clip with full sound right from the first frame, while PGM is recorded. Looking at the cut in the recording using a quality NLE, audio should show up exactly at the start of the first pink frame. Any audio mixer command latency makes the audio show up delayed compared to the first pink frame.

Test material and gear

- Rundown: `Calibration-A6`. See Rundown recipes above.
- A video recorder recording the PGM output. Story Recorder works too.
- Harmonic's Clip Tool, or a serious NLE such as Premiere Pro or Final Cut Pro.

Test

1. On the AV Automation **Genlock** tab, make sure that video server command latency is calibrated and specified.
2. On the AV Automation **Genlock** tab, make sure the switcher command latency value is calibrated and specified.
3. On the AV Automation **Genlock** tab, make sure the audio command latency value is set to 0 ms.
4. Start the video recorder to record the PGM output.
5. Start the rundown and take story by story.
6. Once the rundown has ended, open the recorded video in Clip Tool or an NLE.
7. Find the location in the recording where the silent yellow clip cuts into the pink clip with sine wave sound. The audio should start right at the first pink frame.
8. Using Clip Tool or the NLE, find out where the audio really starts relative to the first pink frame. Do this by ear, stepping through the frames, or visually, by looking at the audio track in the NLE.
9. If the audio is late, increase the audio mixer command latency setting on the AV Automation **Genlock** tab and repeat the test. See Figure 13 below.
10. If the audio is early, decrease the audio mixer command latency setting and repeat the test. See Figure 14 below.
11. If the audio starts at the beginning of the first pink frame, the test is successful.

Repeat the test several times to confirm consistent behavior.

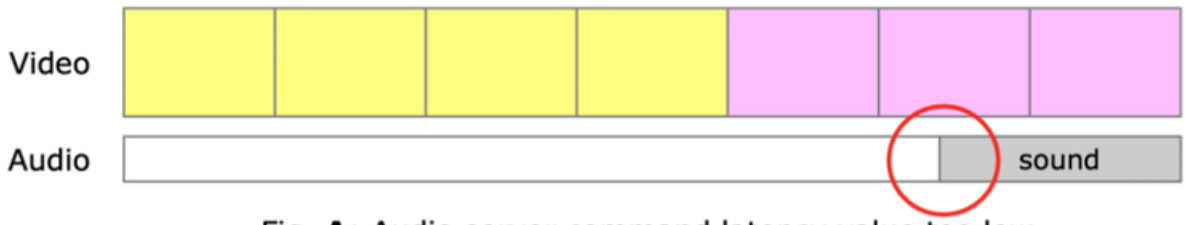


Fig. **A**: Audio server command latency value too low.

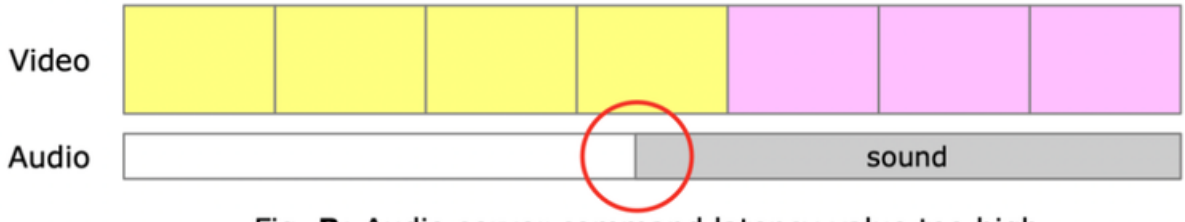
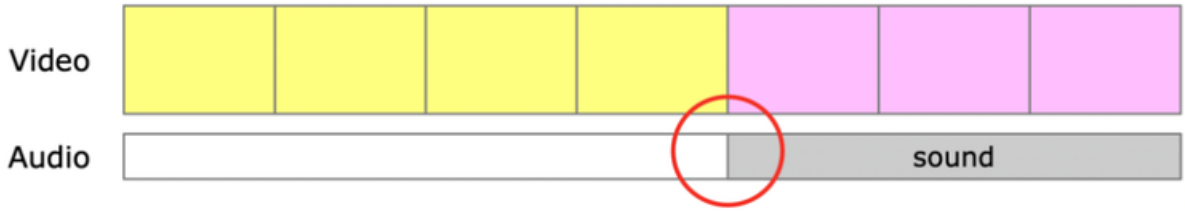
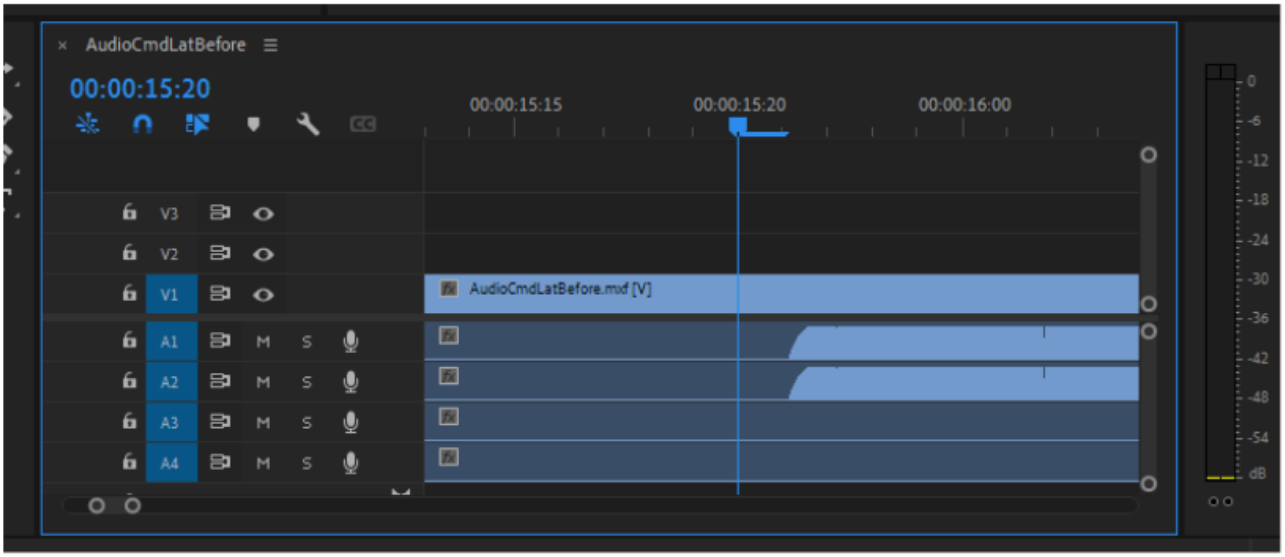
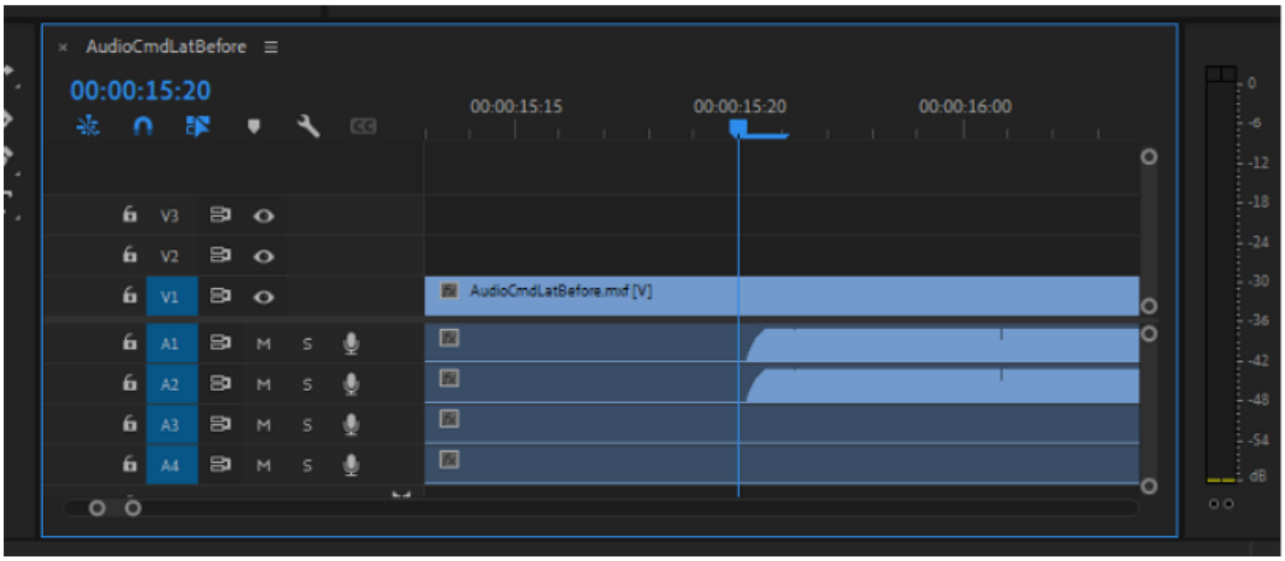


Fig. **B**: Audio server command latency value too high.







11.9.8 B-series calibration tests: alternative tests

Calibration B2: Video server command latency

This test is an alternative to the recommended A2 test.

This test measures the video server command latency, which is the time it takes the video server to respond to a Play command, expressed in frames. Perform the test for each of the individual video server ports involved in the playout of rundown clips.

Test A2 is preferred over this one, because this test depends on the switcher command latency while that value has not been calibrated yet. Instead, this test assumes a switcher command latency of 1 frame, which is a good guess. Use this test if you cannot run test A2, which requires a video recorder.

What is the plan?

Take a yellow clip, followed by a black clip with numbered frames, where the first frame is white. Record the session on a mobile phone from a monitor. Set the video server latency initially very high, which causes Viz Mosart to wait too long for the clip to play. As a result, the video server is already playing well into the clip once it is taken on PGM. The number in the first non-yellow frame in the recording indicates how many frames were waited too long.

Test material and gear

- Rundown: `Calibration-B2`. See Rundown recipes above.
- A mobile phone for recording the PGM output from a monitor.

Test

1. Make sure cabling is restored to normal.
2. On the AV Automation **Genlock** tab, set the video server command latency to 20 frames.
3. On the AV Automation **Genlock** tab, set the switcher command latency to 1 frame.
4. With a mobile phone, record a video of a monitor showing the PGM output.

5. Load and start the rundown. Take story by story until the last one.
6. Once you see the blue numbers on the black background, stop the video recording.
7. On the phone, find the very first non-yellow frame in the recorded video. It shows a number, for example 12. This is the number of frames waited too long for the video server. See Figure 15A below.
8. On the AV Automation **Genlock** tab, set video server command latency to 20 minus the number found. In this example, that results in 8.
9. Reload the rundown and repeat the test.
10. The first non-yellow frame should be a single white frame, followed by black frames, as shown in Figure 15B below. In all other cases the test failed, so retry.

Repeat this test for the other video ports involved in video playout, by changing the templates.

Note: *All these ports should produce the same latency value.*

Having found the latency value, repeat this test a few more times to confirm consistent behavior. In all cases a single white frame followed by black frames should be the result.

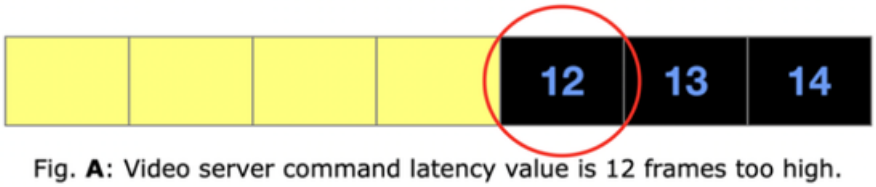


Fig. **A**: Video server command latency value is 12 frames too high.

Fig. **B**: Video server command latency value is perfect now.

Fig. **C**: Video server command latency value is too low.

11.10 Show Design for Frame Accuracy and Story Recorder

- Introduction
- Background
- Definitions
- Show Design Table Columns
- Show Design Table

11.10.1 Introduction

Viz Mosart 5.0 introduced two major features, Story Recorder and frame accurate mode. Neither feature covers the full range of existing Viz Mosart settings, options and workflows. Some of these limits are natural traits of the feature. In frame accurate mode, for example, you no longer have all the time in the world to do things. Other limits exist simply because the work has not been picked up yet.

To design shows with frame accuracy or Story Recorder in mind, you need to know what works and what does not. That is the information provided here.

How to Read the Show Design Table

The show design table serves two audiences, show designers who use frame accuracy without Story Recorder, and show designers who use Story Recorder.

- The **Frame accurate** column applies to anyone running Viz Mosart in genlock mode, whether or not Story Recorder is in use.
- The **SR support**, **SR handling** and **SR backtracking** columns are additional information that only matters when Story Recorder is in use. If you do not run Story Recorder, you can ignore these three columns.

What You Need for Story Recorder

To design a show that works with Story Recorder and gives perfect cuts on a retake, you need:

1. Features marked as supported for Story Recorder, as indicated in the **SR support** column in the show design table below.
2. Features marked as frame accurate, as indicated in the **Frame accurate** column.

In other words, Story Recorder needs frame accuracy, properly set up and calibrated, to produce perfect cuts on retakes for the supported features.

> **Note**: Not supported? If your feature is marked as not supported and you use it anyway, expect mixed and unreliable results. Some things may work fine until you need a retake. Other things may confuse Story Recorder from the start. There is no guarantee.

What You Need for Frame Accurate Performance

This is easy:

1. Features marked as frame accurate, as indicated in the **Frame accurate** column in the show design table.

Note: Not indicated as frame accurate? The feature still works, but it does not execute with predictable, frame accurate precision. Instead, you get the normal Standard way that Viz Mosart has always done things. Whether this is acceptable depends on context.

11.10.2 Background

How Story Recorder Decides What to Record

When activated, Story Recorder sits in the Viz Mosart backend server and closely watches the execution of the rundown story items. Working out what every executed template is supposed to do is virtually impossible, given the flexibility of Viz Mosart, so Story Recorder focuses on the PGM bus of the vision mixer instead. If it sees crosspoint activity in PGM, it also adds the rundown story item just taken to the story item list in the Story Recorder panel.

Much more goes on in the background, but the principle comes down to this:

Story Recorder only cares about story items that affect the PGM video output.

Why a Story Item Does Not Show Up in the Story Recorder Panel

Knowing how Story Recorder thinks, a typical camera or clip story item shows up in the Story Recorder panel when taken. A jingle item that has no crosspoint defined is taken in the rundown, but does not show up in the Story Recorder panel list.

Things go a bit deeper. It is not about the item (template) type, it is about what the template actually does. Some examples:

- A clip template feeding the video signal into an ME other than PP does not show up in the Story Recorder panel, as it does not affect PGM.
- A jingle template that does have a crosspoint defined on PGM shows up in the Story Recorder panel, as it does affect PGM.
- A fullscreen graphics template feeding the video signal into an ME other than PP does not show up in the panel.
- That same graphics template with an accessory added, addressing a PGM crosspoint, does show up in the panel.

As the examples show, Story Recorder does not care about the type of story item or template. It only cares about what the item does on the PGM video bus, and when exactly.

11.10.3 Definitions

These definitions are used throughout this section to clarify the features discussed.

PP and PGM

This section always refers to the virtual PP (Program/Preview). You can map the virtual PP in Viz Mosart to any physical ME, in the **Vision mixer** tab in AV Automation settings.

> **Note:** Story Recorder only monitors activity on the virtual PP, no matter how it is mapped.

PGM always refers to the PGM bus of the virtual PP.

Basic Template and Basic Story Item

A basic template is a bare bones Viz Mosart primary template with minimal configuration, producing the expected content on any ME (including PP) using a crosspoint defined directly in the template itself. No control commands, continue points, directtakes or accessories, and no manual actions.

A basic story item is an item referring to a basic template, without any extras like accessories.

Some examples:

- **Camera/Live**
  - Switcher Crosspoint device enabled and configured to switch to an ME.
  - Fader device or devices enabled and configured with or without crossfade time.
- **Package**
  - Switcher Crosspoint device enabled and configured to switch to an ME.
  - Video Server device enabled with channel configured.
  - Fader device or devices enabled and configured with or without crossfade time.
- **Fullscreen**
  - Switcher Crosspoint device enabled and configured to switch to an ME.
  - Graphics device enabled and configured to play a scene on an engine.
  - Fader device or devices enabled and configured with or without crossfade time.

Graphics Template and Graphics Story Item

A graphics template is a basic template with the graphics device enabled, capable of cutting a graphics device input signal to an ME (including PP), where the crosspoint being switched links directly to a graphics device port, potentially via a ripple.

Similarly, a graphics story item refers to a graphics template, with no accessories as secondaries and no manual actions as directtakes.

Clip Template and Clip Story Item

A clip template is a basic template capable of cutting a video server input signal to an ME including PP, where the crosspoint being switched links directly to a video server port, potentially via a ripple.

Rules for identifying a clip template:

- The template type is a Package, Voiceover or Adlib.
- AND the template is configured to cut to an ME, through the Switcher Crosspoint device using either the `Mix Effect` setting, the `Crosspoint PP` setting or the `Set ME in PP` setting.

- AND the Videoserver device is configured on the template.
- AND the template crosspoint is set to `S_RIP`, or set to a value linked to a video server port configured in the video server configurations from AV Automation.

Take a Voiceover that cuts to a studio situation (camera) and plays a clip on an external device, a monitor in the studio. That Voiceover is not a clip template.

A clip story item refers to a clip template, with no accessories as secondaries and no manual actions as directtakes.

Item1 x Item2

Two story items in a rundown executed back-to-back. For example, a camera followed by a clip.

Hard Cut

Viz Mosart does a hard cut on PGM when it executes a template with `Mix rate` = 0 frames.

11.10.4 Show Design Table Columns

How to interpret values like Yes, No and Limited in the individual table columns.

Frame Accurate

A **Yes** in this column indicates that the feature or setting being discussed is executed aligned to a frame grid towards the controlled device, with proper latency compensation, resulting in deterministic behavior. The controlled device must support frame accurate control for this to work.

A **No** indicates that the feature is not executed frame accurately, but executed similar to how it would run in Standard mode.

A **Limited** indicates that the behavior depends on circumstances, as explained in the associated Comments area.

SR Support

A **Yes** in this column indicates that you can use the feature being discussed in a show design with a Story Recorder workflow in mind.

A **No** indicates that the feature is not supported for the Story Recorder workflow, and you should avoid it in a show design with a Story Recorder workflow in mind. A **No** here can have several reasons:

- Story Recorder may not properly handle the original take.
- The original take may work fine, but Story Recorder cannot do a proper retake.
- The feature may simply not fit the concept of Story Recorder and retakes, for example background audio covering the full show.
- Other possible reasons.

SR Handling

An **Accepted** in this column indicates that the feature or setting results in a new entry being taken in the Story Recorder panel. That is, Story Recorder recognizes the action as something that impacts the PGM bus on virtual PP. A good example is taking a story item with a template that addresses a crosspoint on PGM.

An **Ignored** in this column indicates that Story Recorder does not see the action as something that impacts the PGM bus on virtual PP, so it ignores the take or feature.

An example is taking a camera item followed by a jingle item. Story Recorder accepts the camera item, but not the jingle, since the jingle does not address PGM. The jingle executes as expected in the rundown, but it does not show up as an entry in the Story Recorder panel. To retake the jingle in this example, you first have to retake the previous camera item, then take the jingle again.

A **Depends** is used when the handling depends on a situation that is not necessarily clear, and depends on circumstances explained in the associated Comments area.

A dash (-) indicates Not Applicable.

SR Backtracking

A **Yes** indicates that Story Recorder can restore the original value of the feature being discussed, as part of a retake preroll procedure, going back in time. For example, when you pause the show to retake a previously taken story item, Story Recorder can restore the state of the associated overlay graphics.

A **Limited** indicates that backtracking is supported, but limited in some way. The Comments area explains how.

A **No** indicates that backtracking is not supported. The retake is still possible, but you have to restore the state manually while in Pause mode.

A dash (-) indicates Not Applicable.

11.10.5 Show Design Table

Devices

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Audio mixer	Limited	Yes	-	Yes	Only audio mixers selected by Vizrt, that support frame accurate behavior to begin with, and correctly calibrated, will support frame accurate control with proper latency compensation. Mixers currently known to work frame accurately are: Yamaha (Midimixer), Stagetec.

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
2	Video switcher	Limited	Yes	-	No	Only video switchers selected by Vizrt, that support frame accurate behavior to begin with, and correctly calibrated, will support frame accurate control with proper latency compensation. Switchers currently known to work frame accurately are: TriCaster, Sony Serial Tally.
3	Video server	Limited	Yes	-	Limited	Only video servers selected by Vizrt, that support frame accurate behavior to begin with, and correctly calibrated, will support frame accurate control with proper latency compensation. Video servers currently known to work frame accurately are: Harmonic Spectrum (Omneon). Backtracking supported only for primary items.
4	Overlay graphics	No	Yes	Accepted	Yes	
5	Other devices	No	Yes	-	No	No backtracking supported on subtitling, lights etc.

General

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Basic story item addressing PGM.	Yes	Yes	Accepted	-	The template is addressing the PGM bus only through the Switcher crosspoint device enabled on the template (hard cut or autotrans).

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
2	Any story item addressing PGM only via control commands, continue points, directtakes, accessories or other manual operations.	No	Yes	Accepted	-	
3	Basic story item addressing PGM and also addressing PGM via control commands, continue points, directtakes, accessories or other manual operations.	Yes	Yes	Accepted	-	Story Recorder will respond to the Switcher crosspoint device first and thus the take and retake are frame accurate. The additional PGM crosspoint switches are ignored by Story Recorder.
4	First story item in the show (not a break), regardless whether the PGM is addressed or not.	No	Yes	Accepted	-	

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
5	Break as first item in a show not addressing PGM.	No	Yes	Ignored	-	
6	Break as first item addressing PGM.	No	Yes	Accepted	-	
7	Basic story item addressing an ME, but not PP.	Yes	Yes	Ignored	-	
8	Any story item without switcher crosspoint.	No	Yes	Ignored	-	
9	Any story item addressing PGM via an emem.	Yes	Yes	Ignored	-	Viz Mosart is executing the emem recall frame accurately based on the emem load time given in the template, but Story Recorder does not see what the emem is doing and as such, the template is ignored by SR.

Copyright © 2026 Vizrt

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
10	Item1 x Item2, where: Item1: Clip story item addressing PGM. Item2: Basic story item addressing PGM.	Yes	Yes	Accepted	-	Retaking Item2 will have the preroll done from the original clip and will preserve (redo) the audio and video crossfade, if any.
11	Item1 x Item2, where: Item1: Non-clip, basic story item addressing PGM. Item2: Basic story item addressing PGM.	Yes	Yes	Accepted	-	Retaking Item2 will have the preroll done from the recording and will force a hard cut (no audio and video crossfade).
12	Basic story item addressing PGM with `Hold video transition` (either from UI or as a control command on the template).	No	Yes	Accepted	-	The new item will not show up in the Story Recorder panel until the video transition hold is released again, that is, until the video crosspoint to PGM is executed. This means the take and retake for video are not frame accurate.

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
13	Basic story item addressing PGM with `Hold audio transition` (either from UI or as a control command on the template).	Limited	Yes	Accepted	-	The hold audio transition feature is not relevant for Story Recorder. The take and retake are frame accurate for video, but not audio.
14	Directtakes in any situation.	No	Limited	Limited	No	Here we refer, for example, to directtakes executed manually, through control commands on a template, continue points on template or keyboard shortcuts. The directtakes are not executed frame accurately and they are accepted by Story Recorder depending on what they do. As regarding the backtracking, the directtakes on the templates are executed as part of the retake process, but the targeted devices are not restored beforehand.

Template Settings: General

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Sub-sequence	No	No	Accepted	No	
2	State variants	Yes	No	Accepted	No	

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
3	Accessory as secondary on a basic story item, Pretake is enabled on the accessory template.	No	Yes	Limited[1]	Limited[2]	[1] Story Recording will ignore any crosspoints triggered through the Accessory templates if the story item is already addressing the PGM. However, if the story item is not addressing the PGM while the Accessory template has a crosspoint to switch to PGM, this will be handled by Story Recorder. [2] During a retake situation, the accessory is pretaken during the preroll and executed as part of the retake process, but the targeted devices are not restored beforehand.
4	Accessory as secondary on a basic story item, Preload is enabled on the accessory template.	No	Yes	Limited[1]	Limited[2]	[1] Story Recording will ignore any crosspoints triggered through the Accessory templates if the story item is already addressing the PGM. However, if the story item is not addressing the PGM while the Accessory template has a crosspoint to switch to PGM, this will be handled by Story Recorder. [2] During a retake situation, the accessory is preloaded during the preroll and executed as part of the retake process, but the targeted devices are not restored beforehand.

Template Settings: Control Commands and Continue Points

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Control commands on take, continue points, on take-out on a template.	No	Limited	Limited	No	Control commands are re-executed as part of the retake process, but the targeted devices are not restored beforehand.
2	Autotake template with very short duration and too many autotake continue points.	No	No	Accepted	-	Every autotake continue point will claim approximately 1-2 seconds of delay in the Viz Mosart backend. If the sum of this delay exceeds the duration of the story item, the item will not be auto taken at the expected point in time.

Template Settings: Switcher Crosspoint

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Basic story item with `Mix Delay` >= 0 on the template and `Transition Only` enabled.	Yes	Yes	Accepted	-	

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
2	Basic story item with `Mix Delay` > 0 on the template and `Transition Only` disabled.	Yes	Yes	Accepted	-	
3	Basic story item with `Mix Delay Out` > 0.	Yes	Yes	Accepted	-	
4	Basic story item addressing PGM using `Set ME in PP` set in the template's Switcher crosspoint device.	Yes	Yes	Accepted	-	This setting is taken into consideration only if `Mix Effect` is not PP on the Switcher Crosspoint device of the template.
5	Basic story item addressing PGM using `Crosspoint PP` set in the template's Switcher crosspoint device.	Yes	Yes	Accepted	-	This setting is taken into consideration only if `Mix Effect` is not PP and `Set ME in PP` is not enabled on the Switcher Crosspoint device of the template.

Copyright © 2026 Vizrt

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
6	Basic story item addressing PGM, but `Crosspoint Hold` is set in the template's Switcher crosspoint device.	-	Yes	Ignored	-	The crosspoint is not executed because of the `Crosspoint Hold` setting.

Template Settings: Emem Recall

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Basic story item addressing PGM with Emem recall.	Yes	Yes	Accepted	No	

Template Settings: Macro Recall

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Basic story item addressing PGM + Recall macro not addressing PGM.	Yes	Yes	Accepted	No	

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
2	Basic story item not addressing PGM + Recall macro addressing PGM.	No	Yes	Ignored	No	

Template Settings: KeyFill / Auxiliaries

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	KeyFill/ Auxiliary (key set delay >= 0)	No	Yes	-	No	

Template Settings: Video Server

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Clip story item with `Loop` enabled on the template's Videoserver device.	Yes	No	Accepted	-	Story Recorder cannot reproduce the original crossfade with looping clips.

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
2	Adlib clip, resumed from a paused position within the story.	Yes	No	Accepted	No	Adlibs pausing and resuming the same clip over several items within the same story is not supported for Story Recorder. There is no backtracking to the correct position of the adlib clip when doing a retake.

Template Settings: Graphics (Fullscreen Graphics)

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Item1 x Item2. Item1: Basic story item without Graphics device. Item2: Basic story item with Graphics device.	Yes	Yes	Accepted	No	This is the standard approach for (fullscreen) graphics where the graphics device is cued and started behind a closed crosspoint, followed by opening the crosspoint in PGM. The take on the engine is done slightly early, and thus a few frames are lost, but this has always been the case. Note that backtracking for fullscreen graphics is not supported.
2	Item1 x Item2, a/b roll for graphics. Item1: Basic story item with Graphics device, a/b roll enabled. Item2: Basic story item with Graphics device, a/b roll enabled.	Yes	Yes	Accepted	No	A variation of the above, both graphics channels have their own dedicated crosspoint addressing PGM.

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
3	Item1 x Item2, same graphic channel. Item1: Basic story item with Graphics device addressing a graphic engine (channel). Item2: Basic story item with Graphics device, addressing same graphic engine (channel) as above.	No	No	Accepted	No	This refers to the switch of graphics on the same channel of the graphics device, on an open crosspoint. The switch will show a few frames early, making proper retakes impossible. **NOT RECOMMENDED**. Workarounds: (1) Retake from first Fullscreen. (2) Use graphics ripple. (3) Manually restore first graphic on the engine when retaking the second graphic.
4	Animated fullscreen graphics.	No	No	Accepted	No	No backtracking supported on fullscreen graphics. In a retake situation, the animation will not be cued properly during preroll.

Template Settings: Video Wall

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Video Wall	No	Yes	-	No	

Template Settings: Audio Player

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Item1 x Item2. Item1: Basic story item (non-clip) with Audio Player device. Item2: Basic story item.	Limited	Yes	Accepted	-	The audio file is not played frame accurately. The preroll is done from recording and as such no backtracking for audio files is needed.
2	Item1 x Item2. Item1: Clip story item with Audio Player device. Item2: Basic story item.	Limited	No	Accepted	No	The audio file is not played frame accurately. Audio player files are not properly cued during preroll, resulting in a discontinuity at the retake. The audio files are not backtracked.

Template Settings: Virtual Set

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Virtual Set	No	Yes	-	No	

Template Settings: Camera Control

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Camera control	No	Yes	-	No	

Template Settings: GPI/O Control

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	GPI/O Control	No	Yes	-	No	

Template Settings: Light Control

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Light Control	No	Yes	-	No	

Template Settings: Router Control

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Router control	No	Yes	-	No	

Template Settings: Weather

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Weather	No	Yes	-	No	

Template Settings: Audio Mixer

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Audio mixer recall	No	Yes	-	No	

Template Settings: Audio Faders

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Audio faders	Yes	Yes	-	Yes	

AV Automation Device Properties: Vision Mixer Settings

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Virtual to Physical ME mapping	-	Yes	-	-	Story Recorder only monitors PGM on virtual PP to make it appear in the EDL list.

Copyright © 2026 Vizrt

AV Automation Config: Effects

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Basic story item addressing PGM through an effect with `Emem Nr. Program` defined or `Macro Program Action` set.	Yes	Yes	Accepted	-	The template's crosspoint will be set on preview and the effect will trigger the switcher to do the autotrans. Note: For the retake to work properly, the template should also have the standard crosspoint defined to address the PGM (for the cases where a hard cut is enforced).
2	Non-clip basic story item with effects with duration.	Yes	Yes	Accepted	-	
3	Clip story item with effects with duration.	Limited	Yes	Accepted	-	The clip is started non-frame accurately.

Features: Audio

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Audio clip exceeding the boundaries of an item (for example, a clip played during the entire story or show).	No	No	-	No	No retake can be done in the story or show as there will always be a "jump" in the audio.

Features: Overlays

#	Feature	Frame accurate	SR support	SR handling	SR backtracking	Comments
1	Rundown overlay graphics	No	Yes	Accepted	Yes	
2	Named overlays taken manually	No	Yes	Accepted	Yes	
3	Manual overlays (rundown or keyboard shortcuts)	No	Yes	Accepted	Yes	
4	Overlays executed through control commands on a template	No	Limited	Accepted	Limited	
5	Animated overlays (for example, moving tickers)	No	No	Accepted	Limited	The animated graphics are backtracked as any overlay, but they cannot be cued properly during preroll, resulting in discontinuity at the take.

Copyright © 2026 Vizrt

11.11 PTP Setup

- PTP Setup
- Requirements
- Supported Deployment Scenarios
- Setting up the Windows PTP Provider
- Viz Mosart Configuration
- Verification
- Troubleshooting

11.11.1 PTP Setup

Viz Mosart can derive genlock and timecode from a PTP (Precision Time Protocol) Grandmaster instead of a traditional hardware timecode reader such as a Plura PCIe card. The Windows system clock is disciplined to the PTP Grandmaster through the built-in W32Time PTP provider (`ptpprov.dll`). Viz Mosart then uses the SMPTE ST 2059-1 epoch to calculate frame boundaries, so no additional hardware is needed.

For the frame accuracy concepts, and for the calibration that Viz Mosart needs whichever clock source you choose, see Frame Accurate System Operations.

11.11.2 Requirements

Windows Version

The PTP provider (`ptpprov.dll`) is available on:

- Windows 10 version 1809 (October 2018 Update) and later
- Windows 11, all versions
- Windows Server 2019 and later

Verify that the file exists:

```
Test-Path C:\Windows\System32\ptpprov.dll
```

Network

- UDP ports 319 and 320 must be open (inbound) on the Viz Mosart machine firewall.
- The Viz Mosart machine must have network connectivity to the PTP Grandmaster, or to a PTP boundary clock on the same subnet.
- PTP multicast traffic (`224.0.1.129`) must be able to reach the NIC on the Viz Mosart machine.
- If the machine has more than one network adapter, make sure PTP multicast reaches the NIC on the same network as the PTP Grandmaster. PTP is typically carried on the media or production network, not the management network.

PTP Grandmaster

A PTP Grandmaster must be operational on the network. You need to know:

- The IP address of the PTP Grandmaster, or of a boundary clock reachable from the Viz Mosart machine.
- The PTP domain number used by the Grandmaster. The SMPTE ST 2059-2 default is `127`, but many facilities use a different value.

Viz Mosart Version

PTP support requires Viz Mosart 5.15.0 or later.

11.11.3 Supported Deployment Scenarios

Scenario	Viz Mosart runs on	PTP is configured on	Notes
Physical machine	The machine itself	Same machine (`ptpprov.dll`)	Standard setup. Covered in this section.
VMware with host PTP	Guest VM on ESXi	ESXi host (Precision Clock)	ESXi passes accurate time to the guest. See the VMware documentation.
VMware with guest PTP	Guest VM on ESXi	Inside the guest (`ptpprov.dll`)	Same as physical. Disable VMware time sync to avoid conflicts.
Hyper-V	Guest VM on Windows Server	Inside the guest (`ptpprov.dll`)	Same as physical. Disable Hyper-V time sync to avoid conflicts.

The rest of this section covers the physical machine and in-guest `ptpprov.dll` setup, which applies to all scenarios except VMware with host PTP.

> **Tip:** VMware host PTP setup is recommended for virtual machines. ESXi 7.0 Update 3 and later supports SMPTE ST 2059-2 natively. To use this path, configure PTP on the ESXi host, add a Precision Clock virtual device to the Viz Mosart VM (requires VM hardware version 17 or later), and the Windows guest then receives accurate time from the host automatically. With this approach, PTP multicast does not need to reach the virtual NIC of the VM, because the host handles PTP and the guest reads the virtual clock device. Continue at Viz Mosart configuration below once the VMware setup is complete.

Copyright © 2026 Vizrt

11.11.4 Setting up the Windows PTP Provider

Note: The steps below are based on the official Microsoft W32Time PTP client configuration. See the References section at the end of this page.

Run all commands in an elevated PowerShell session (Run as Administrator).

Step 1: Verify that ptpprov.dll is present

```
Test-Path C:\Windows\System32\ptpprov.dll
```

This must return `True`. If it returns `False`, the Windows version does not include the PTP provider. See Requirements above.

Step 2: Open the firewall ports

```
New-NetFirewallRule `
    -DisplayName 'PTP Event (UDP 319)' `
    -Name 'PTP-319' `
    -LocalPort 319 `
    -Direction Inbound `
    -Action Allow `
    -Protocol UDP

New-NetFirewallRule `
    -DisplayName 'PTP General (UDP 320)' `
    -Name 'PTP-320' `
    -LocalPort 320 `
    -Direction Inbound `
    -Action Allow `
    -Protocol UDP
```

Note: If you see "Cannot create a file when that file already exists", the rules are already in place and the message is safe to ignore.

Step 3: Register the PTP provider

Replace `<GM_IP>` with the IP address of the PTP Grandmaster or boundary clock on your local subnet. Replace `<DOMAIN>` with the PTP domain number, which is `127` by default.

```
$ptp = "HKLM\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\PtpClient"
```

```
reg add $ptp /v DllName          /t REG_SZ    /d "c:\windows\system32\ptpprov.dll" /f
reg add $ptp /v Enabled           /t REG_DWORD /d 1 /f
reg add $ptp /v InputProvider     /t REG_DWORD /d 1 /f
reg add $ptp /v PtpMasters        /t REG_SZ    /d "<GM_IP>" /f
reg add $ptp /v DomainNumber      /t REG_DWORD /d <DOMAIN> /f
reg add $ptp /v EnableMulticastRx /t REG_DWORD /d 1 /f
reg add $ptp /v DelayPollInterval /t REG_DWORD /d 0x3e80 /f
reg add $ptp /v AnnounceInterval  /t REG_DWORD /d 0x0fa0 /f
```

Note: The PTP domain number must match the Grandmaster. If it is wrong, W32Time silently ignores all PTP packets and never synchronizes.

Understanding PtpMasters. The Windows PTP provider is a listen-only client. It receives PTP multicast but sends no outbound PTP traffic. `PtpMasters` is a whitelist of source IP addresses: the provider accepts PTP Sync and Announce packets only from IP addresses in this list, and ignores all others. The address does not have to be the actual PTP Grandmaster. In most broadcast facilities, a PTP boundary clock, often the PTP-aware network switch that also serves as the subnet gateway, redistributes time from the Grandmaster to each subnet. Set `PtpMasters` to the IP address of whichever device sends PTP multicast on your local network.

How the three critical settings work together. PTP multicast packets arrive at the NIC regardless of these settings, because they flow on the network at layer 2 and 3. However, `ptpprov.dll` does not process them unless all three of the following conditions are met:

1. `EnableMulticastRx` is `1`, which enables the provider to listen for multicast PTP.
2. `PtpMasters` contains the source IP address, because the provider accepts PTP only from whitelisted addresses.
3. `DomainNumber` matches the PTP domain, because the provider accepts PTP only on the configured domain.

If any one of these is wrong, PTP fails silently and W32Time reports "Local CMOS Clock".

Verify that the registry was written correctly:

```
reg query $ptp
```

Step 4: Disable competing time providers

Disable NTP and the Hyper-V or VMware time integration service (VMIC) so that they do not compete with PTP. This matches the Microsoft reference configuration for a PTP client.

```
# Disable NTP client
$ntp = "HKLM\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpClient"
reg add $ntp /v Enabled /t REG_DWORD /d 0 /f

# Disable VMIC time provider (safe to run on bare metal)
$vmic =
"HKLM\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\VMICTimeProvider"
```

```
reg add $vmic /v Enabled /t REG_DWORD /d 0 /f
```

If NTP or VMIC is still enabled and has a valid source, W32Time may choose it over PTP. The Microsoft reference configuration disables both.

On virtual machines, you may also need to disable time synchronization at the hypervisor level:

- VMware: disable **Synchronize guest time with host** in the VMware Tools settings.
- Hyper-V: disable **Time synchronization** in Integration Services, or run:

```
Disable-VMIntegrationService `
    -VMName "<VMName>" `
    -Name "Time Synchronization"
```

Step 5: Restart W32Time

```
net stop w32time
net start w32time
w32tm /resync
```

Step 6: Verify the PTP lock

```
w32tm /query /status
```

Example of a successful PTP lock:

```
Leap Indicator: 0(no warning)
Stratum: 3 (secondary reference - syncd by (S)NTP)
Precision: -23 (119.209ns per tick)
Root Delay: 0.0001500s
Root Dispersion: 0.0100020s
ReferenceId: 0x4D505450 (source IP:  77.80.84.80)
Last Successful Sync Time: 1/15/2026 10:30:45 AM
Source: 192.168.1.100
Poll Interval: 10 (1024s)
```

What to check:

Field	Expected	Meaning
ReferenceId	`0x4D505450`, shown as `77.80.84.80`	ASCII "MPTP". This confirms PTP.

Field	Expected	Meaning
Source	The Grandmaster IP address, for example `192.168.1.100`	Shows an IP address, not "PTP". `Local CMOS Clock` or `Free Running` means PTP is not active.
Stratum	A low number, not 0	Each hop from the Grandmaster adds 1. See the note below.
Last Successful Sync Time	A recent timestamp	Confirms active synchronization.

About Stratum. PTP and NTP use a hierarchy of time sources. A reference clock such as a GPS receiver or atomic clock sits at stratum 0. Any clock synchronized with a stratum 0 clock becomes stratum 1, and so on. The stratum number indicates distance from the reference clock, not the accuracy or reliability of the time. Each boundary clock in the chain adds one level:

- Stratum 1: direct sync to a stratum 0 source such as a GPS or atomic clock. Rare.
- Stratum 2: syncing to a PTP Grandmaster, which is stratum 1.
- Stratum 3: syncing through one boundary clock. This is the common case.
- Stratum 4 and above: syncing through multiple boundary clocks.
- Stratum 0, reported as unspecified: the PTP provider never engaged. Source shows `Local CMOS Clock`. Re-check Step 3 above.

Note: Ignore "(S)NTP" in the Stratum line. W32Time always labels itself as "(S)NTP" regardless of the actual time source, so even when PTP is active the output reads `syncd by (S)NTP`. This is a display quirk. The `ReferenceId` field is the only reliable way to confirm PTP discipline.

If the status shows `Free Running`, `Local CMOS Clock`, or Stratum 0, see Troubleshooting below.

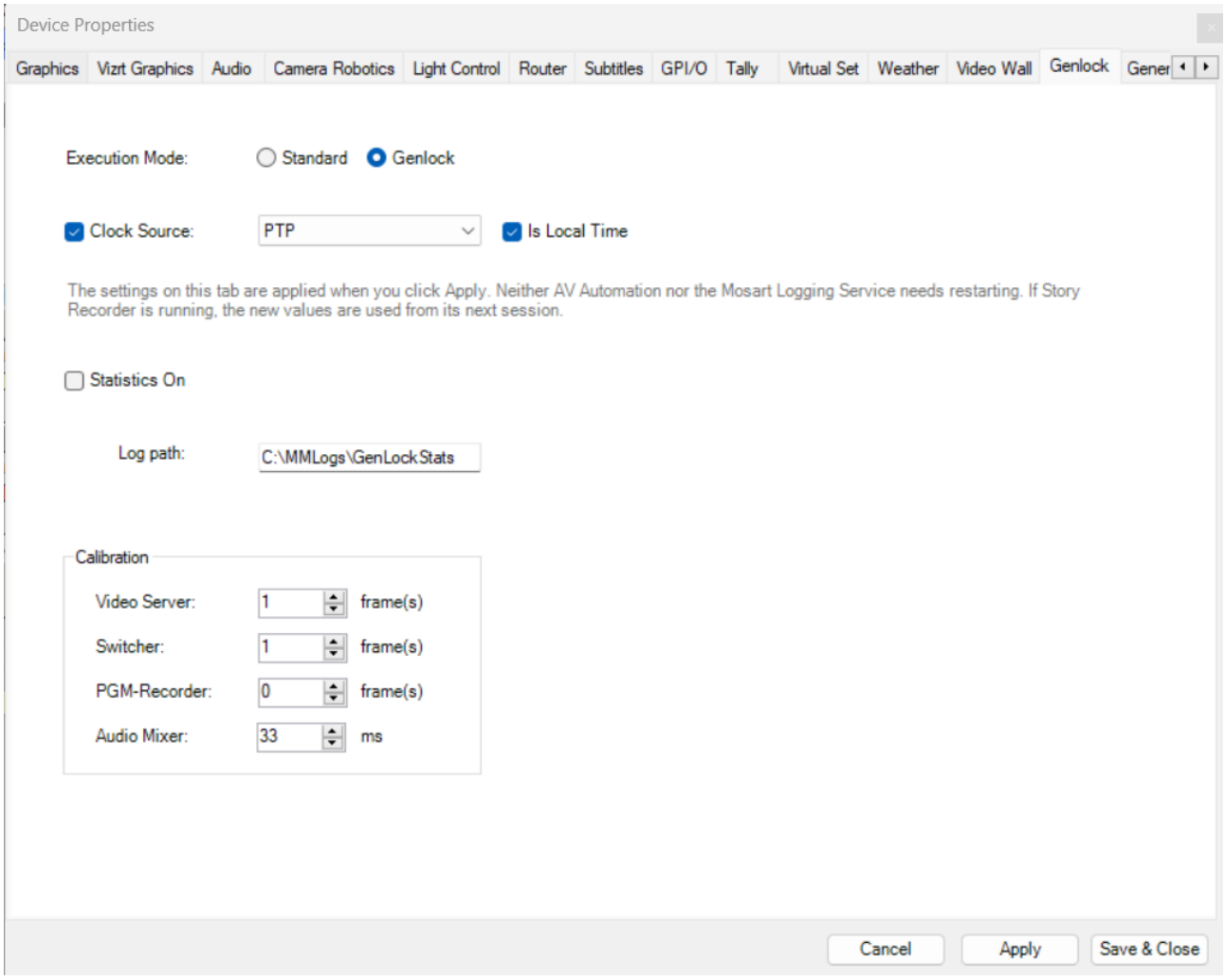
11.11.5 Viz Mosart Configuration

Once the Windows system clock is PTP-disciplined, confirmed with `w32tm /query /status`, configure Viz Mosart.

Setting	Value	Notes
Clock Source	`PTP`	Selects PTP as the clock source.
Enable Genlock	`True`	Must be enabled for PTP to function.

Setting	Value	Notes
External clock is local time	`True` or `False`	`True` means timecode represents local wall-clock time. `False` means UTC. This must match the convention used by your video server and recorder. If set to `True`, also enable **Use Local Time** in Manus Settings, which Story Recorder requires.
Frame rate	Must match the PTP Grandmaster	For example 25 fps or 29.97 fps. A mismatch causes timecode drift. Frame rate is configured in Manus Settings, not in AV Automation.

Clock Source, Enable Genlock, and External clock is local time are set in AV Automation. After changing Clock Source, restart AV Automation for the change to take effect.

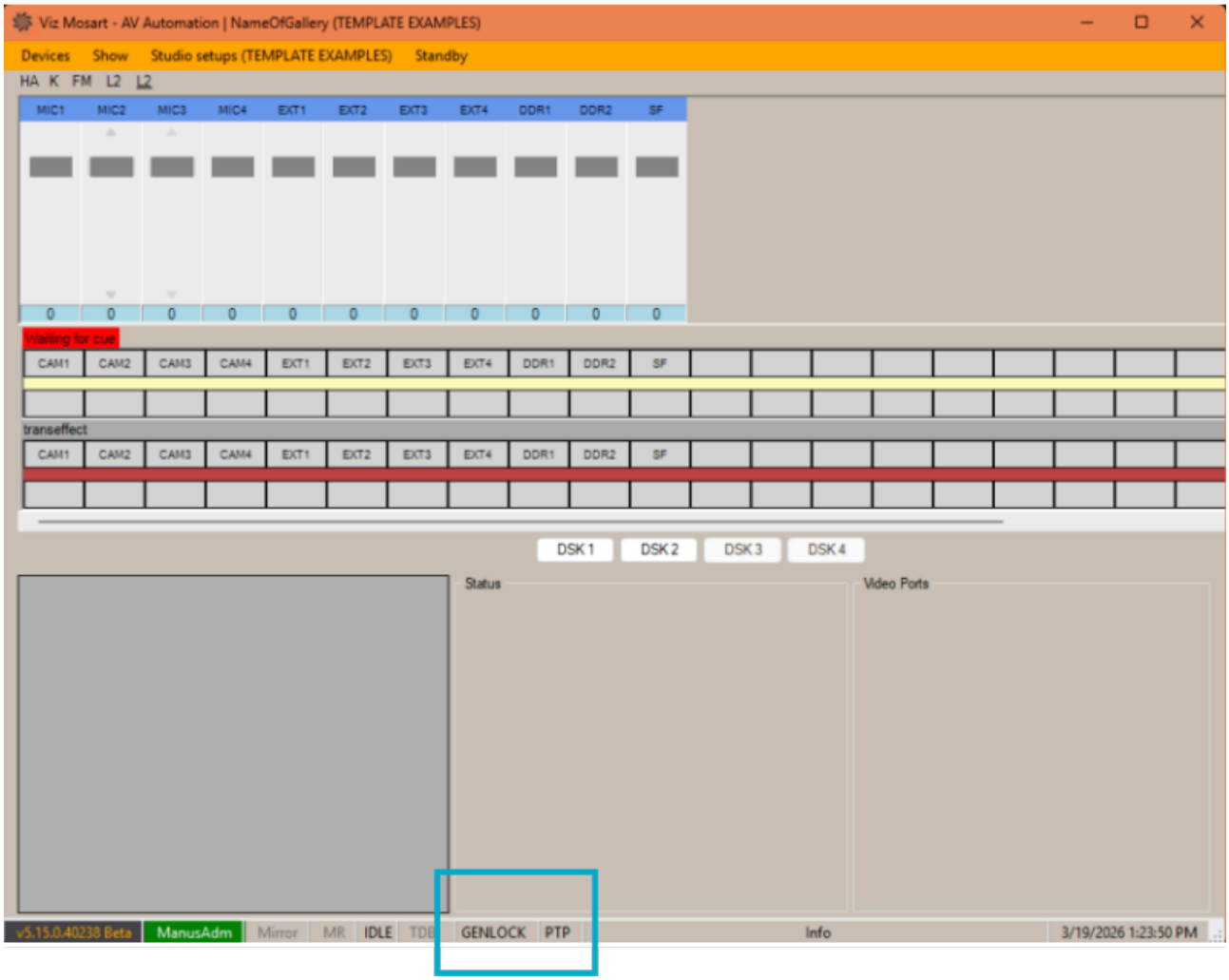


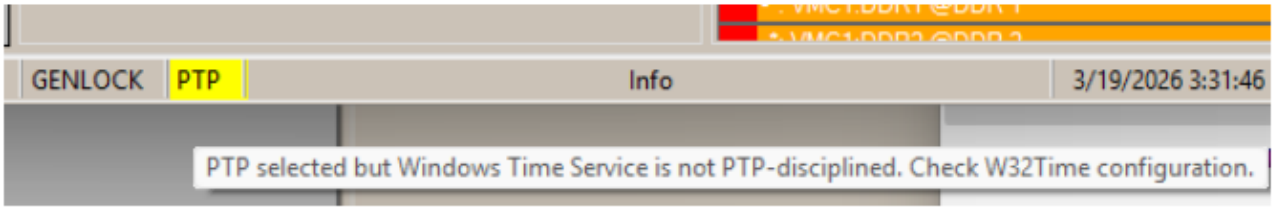
11.11.6 Verification

Viz Mosart Status Bar

The genlock status bar in AV Automation indicates the current state:

Color	Meaning
Orange	Viz Mosart is validating the PTP clock source. This is normal at startup, because it requires 100 consecutive good frames, about 4 seconds at 25 fps. The tooltip shows progress.
No color, the default gray	PTP is locked and running normally.
Yellow	PTP is running but something needs attention, for example W32Time lost PTP discipline or jitter is elevated. Check the tooltip for details.
Red	The PTP clock source failed to start or is unhealthy. Check `w32tm /query /status`.





Genlock Statistics

1. Press **Ctrl+Shift+I** in AV Automation to start collecting genlock statistics. The status bar shows "PTP - STATISTICS ON" while collection is active.
2. Collection stops on its own once 10,000 samples have been gathered. Viz Mosart takes one sample per frame, so how long that takes depends on the frame rate: about 6 to 7 minutes at 25 fps, about 5 and a half minutes at 29.97 fps, and roughly half those times at 50 and 59.94 fps.
3. Press **Ctrl+Shift+I** again to stop earlier and write the results.
4. Three files are created in `C:\MMLogs\GenLockStats\` by default:

- `GenLockStats_Summary_<uid>.txt`, a human-readable summary with conclusions.
- `GenLockStats_FrameClock_<uid>.csv`, the internal frame execution timing and jitter.
- `GenLockStats_ExternalCS_<uid>.csv`, the raw PTP clock source timing, phase shift, and timecode.

5. Send all three files to Vizrt Support.

Key items to check in the report:

- **Clock source type** should show `PTP (SMPTE ST 2059-1)`.
- **W32Time PTP Health** shows the sync source IP address, stratum, phase offset, and Reference ID. A Reference ID of `0x4D505450` confirms PTP discipline.
- **Jitter** should be consistently below 2 ms, and is typically within 0.1 ms.
- **Bad frames** should be 0 during stable operation.
- **PTP Clock Quality** in the conclusion section rates the overall quality. "GOOD" means all metrics are healthy.

Checking the PTP Offset Over Time

To monitor the offset between the local clock and the PTP Grandmaster:

```
w32tm /stripchart /computer:<GM_IP> /samples:10 /dataonly
```

Values should be consistently small, below 1 ms. Include this output when contacting Vizrt Support about timing issues.

Continuous Monitoring

Viz Mosart checks `w32tm /query /status` every 60 seconds to verify that the system clock is still PTP-disciplined. If PTP discipline is lost, for example through a network problem or the Grandmaster going offline, Viz Mosart detects this and shows a warning in the status bar within 60 seconds.

11.11.7 Troubleshooting

The PTP Provider is not Active (Free Running or Local CMOS Clock)

The troubleshooting commands below use variables for the long registry paths:

```
$ptp = "HKLM\SYSTEM\CurrentControlSet\Services\W32Time" + `
       "\TimeProviders\PtpClient"
$ntp = "HKLM\SYSTEM\CurrentControlSet\Services\W32Time" + `
       "\TimeProviders\NtpClient"
```

1. **Is** `ptpprov.dll` registered? Verify the registry with `reg query $ptp`. If values are missing, run Step 3 again.
2. **Is the PTP Grandmaster IP address correct?** Check with `reg query $ptp /v PtpMasters`.

3. **Is the PTP domain number correct?** This is the most common problem, because the domain must match the Grandmaster. Check with `reg query $ptp /v DomainNumber`.
4. **Is multicast reception enabled?** It must be `0x1`. Check with `reg query $ptp /v EnableMulticastRx`.
5. **Is the firewall blocking PTP?** Check with:

```
Get-NetFirewallRule -DisplayName 'PTP*' |
    Select-Object Name, Enabled, Direction, Action
```

6. **Is NTP still enabled and overriding PTP?** Check with `reg query $ntp /v Enabled`. The value should be `0x0`.
7. **Is W32Time running?** Check with `sc query w32time`.

Stratum 0 or Local CMOS Clock, Meaning the Provider never Engaged

Stratum 0 with `Source: Local CMOS Clock` means `ptpprov.dll` never engaged and the PTP provider is not properly configured. Verify:

- Network connectivity, using `ping <GM_IP>`.
- That the PTP domain number matches the Grandmaster.
- That PTP multicast (`224.0.1.129`) can reach the machine. Check the NIC and VLAN configuration.
- That UDP ports 319 and 320 are not blocked.

The Status Bar shows Yellow and Reports "not PTP-disciplined"

Viz Mosart checks `w32tm /query /status` every 60 seconds. If the `ReferenceId` is not `0x4D505450`, a yellow warning appears. Fix the Windows PTP configuration first, and Viz Mosart detects the lock automatically within 60 seconds.

The Status Bar shows Red

The PTP clock source failed to start. Check:

- That `w32tm /query /status` shows a valid PTP lock, with `ReferenceId` `0x4D505450`.
- That Clock Source is set to `PTP` and Enable Genlock is `True` in AV Automation.
- That AV Automation was restarted after the settings were changed.

Timecode Mismatch between Viz Mosart and the Recorder

1. **Frame rate.** The configured frame rate in Viz Mosart must match the frame rate of the PTP Grandmaster. A mismatch causes progressive drift.
2. **UTC against local time.** If the recorder uses local time and Viz Mosart uses UTC, or the other way round, a fixed offset results. Adjust the **External clock is local time** setting.
3. **Drop-frame.** For 29.97 fps and 59.94 fps, make sure both Viz Mosart and the recorder agree on drop-frame timecode.

No PTP Packets reach the Machine

To verify that PTP traffic is arriving, use the built-in Windows packet monitor:

```
pktmon filter remove
pktmon start --etw -m real-time
```

Look for lines containing `224.0.1.129` and port `319` or `320`. Press **Ctrl+C** to stop the capture. If PTP packets appear, traffic is flowing correctly.

> ***Tip:*** *Use an unfiltered capture. The pktmon UDP port filter does not reliably match multicast PTP packets.*

If you see PTP packets followed by `Drop: DropReason INET: transport endpoint was not found`, this is normal before W32Time is configured. It means PTP traffic reaches the machine but nothing is listening on that port yet. Once `ptpprov.dll` is registered and W32Time is restarted, the drops stop.

If no PTP packets appear:

- Verify that you are monitoring the correct NIC. PTP is typically on the media network, not the management network.
- Check that the switch forwards PTP multicast to this port, which depends on the IGMP snooping configuration.
- Confirm that the Grandmaster is on the same subnet or VLAN.

References

- Microsoft W32Time PTP documentation
- Microsoft: how to set up a Precision Time Protocol client
- VMware vSphere 8: use PTP servers for host time synchronization
- VMware vSphere 8: add a Precision Clock device to a virtual machine
- SMPTE ST 2059-2:2021, SMPTE profile for use of IEEE 1588

Copyright © 2026 Vizrt

12 Operational Examples

Viz Mosart can be configured to support multiple specialist scenarios. In this section, we list individual workflows that showcase the flexibility of Viz Mosart.

- Device-Specific Operations
- Media Sequencer
- Mosart Remote Control REST API
- Story Recorder

12.1 Device-Specific Operations

This section lists setup and operation for the following devices or products:

- Viz Arc support within a Mosart show
- Viz Trio support in a Viz Mosart Show
- Viz Engine as a Switcher
- Avid iNews Web Service for Status Feedback
- Jupiter Systems PixelNet Video Wall Control
- Lama Mix and Lama Connect
- Panasonic Kairos Vision Mixer
- Stagetec Driver Configuration
- Stream Deck
- TriCaster and Viz Mosart
- Flowics and Viz Mosart
- Adaptive Live Producer

Tip: For other devices not listed here, refer to the AV Automation topic, *Device Properties*.

Copyright © 2026 Vizrt

12.1.1 Viz Arc support within a Mosart show

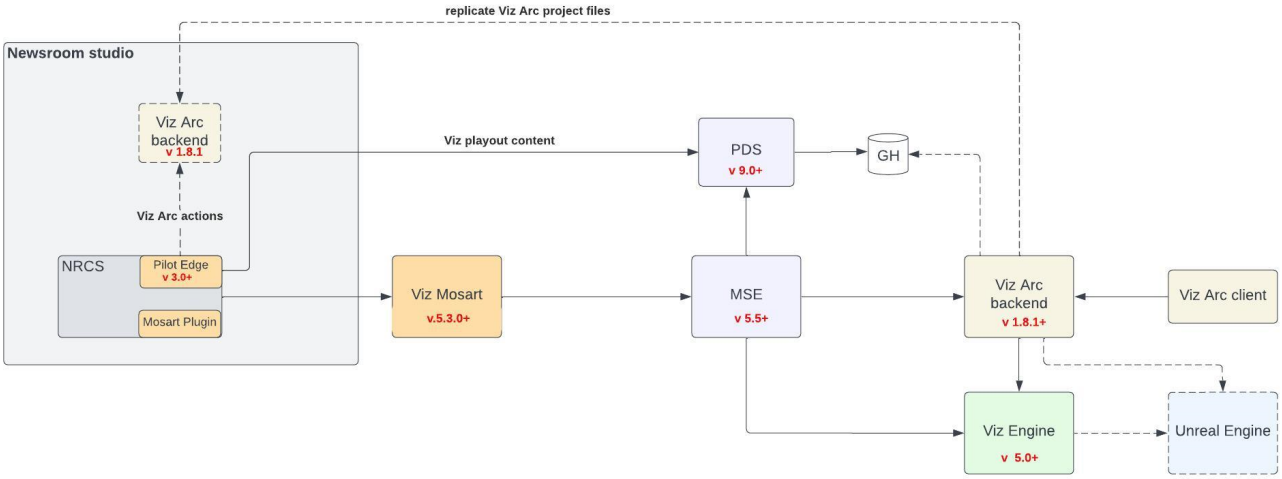
Viz Arc is an augmented reality control system, which enables users to drive virtual sets, AR graphics, Viz Engine and Unreal® Engine graphics from a single interface. These elements can also be driven from a Viz Mosart controlled show.

- Requirements
- MOS integration
- Driver Configuration
- Viz Mosart Workflow

Configuration and Setup

Requirements

The illustration below highlights the main actors and minimum component versions required for configuring Viz Mosart to manage Viz Arc actions within an NRCS-created rundown.



MOS integration

To setup Viz Pilot Edge with Viz Arc

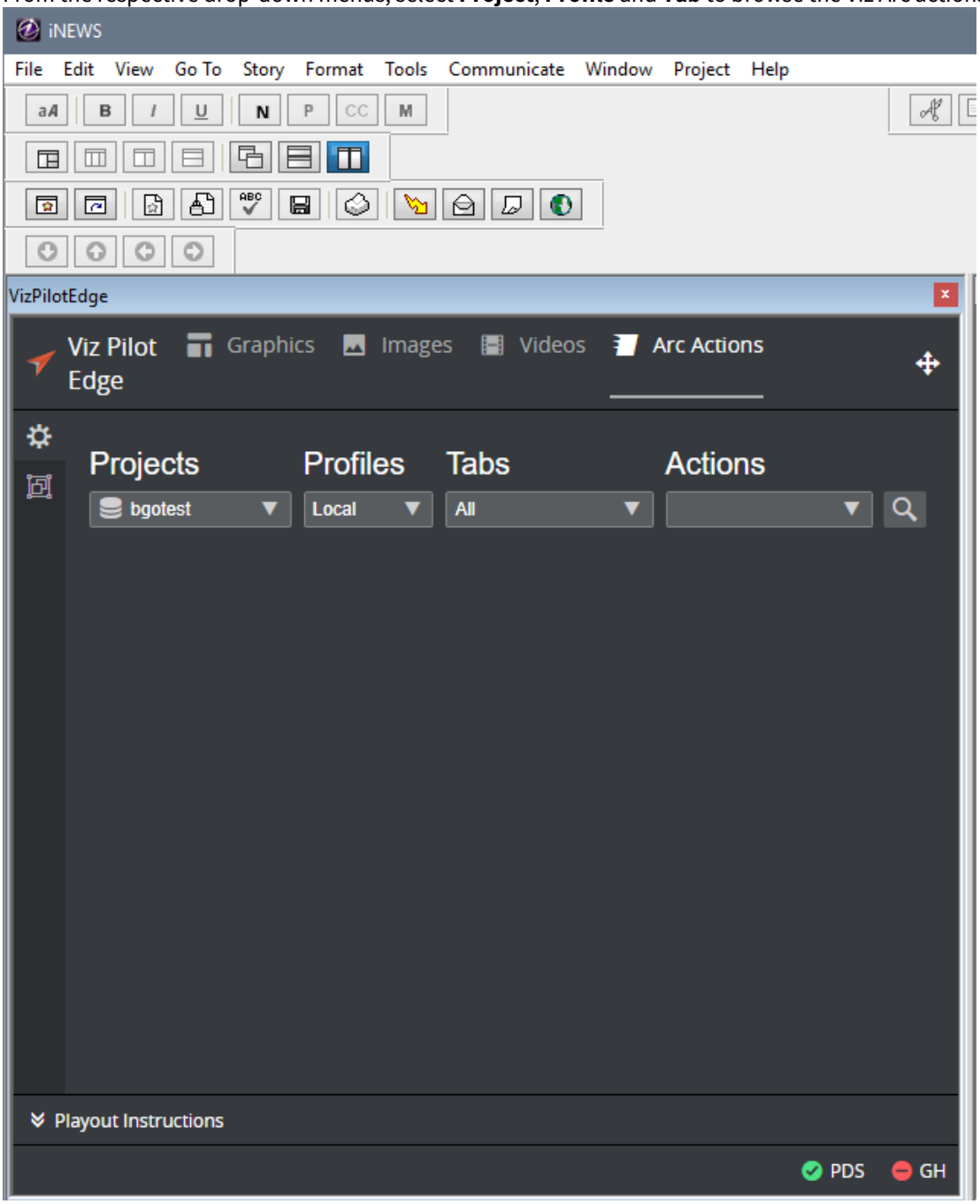
- Refer to the Viz Pilot Edge documentation.

Note: In the diagram above

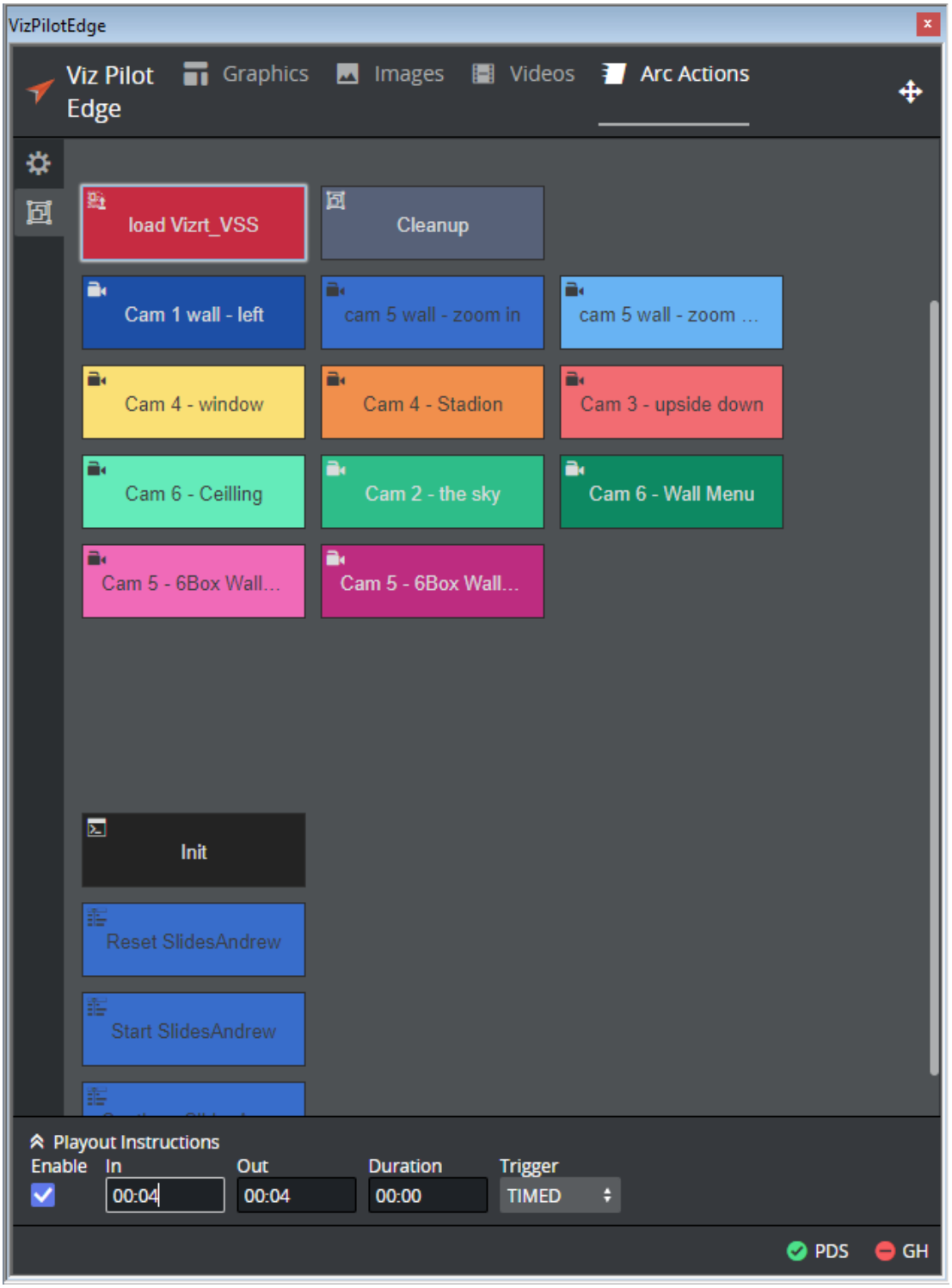
- A *second* Viz Arc backend is set up on the Newsroom System machine for security reasons.
- Viz Arc does not currently support authentication. If this is not an issue, Viz Pilot Edge can be set up to directly fetch data from the Viz Arc playout backend.

To create a Mosart show containing pre-planned or manual Viz Arc actions

1. Open Viz Pilot Edge from the NRCS.
2. From the respective drop-down menus, select **Project**, **Profile** and **Tab** to browse the Viz Arc actions.



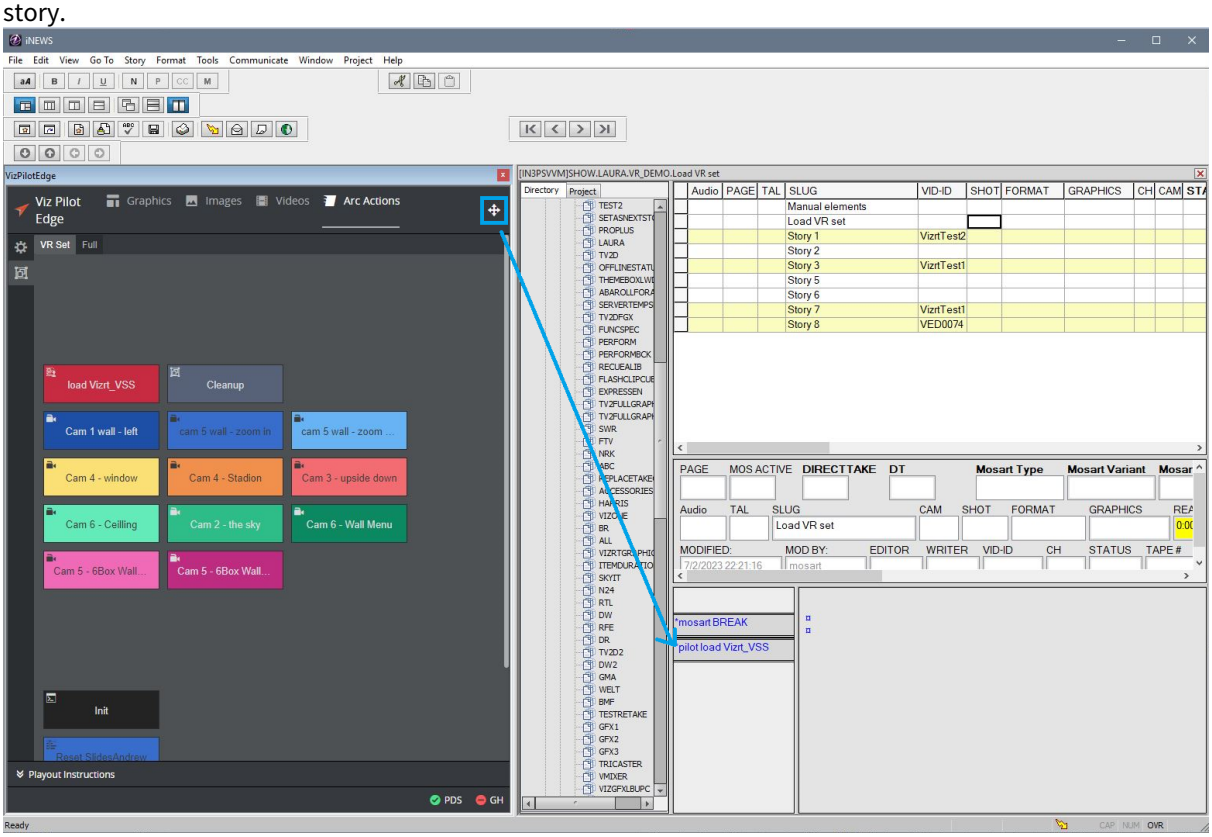
3. Select a Viz Arc action and enable the *Playout Instructions* to add timing and the trigger (Timed or Manual).



4. When the trigger is set to *TIMED*, the **In** field can be set to give the offset in minutes and seconds relative to the story item start.
    a. In this case, Viz Mosart will play the action when the story item is taken or with the given offset.
    b. If the Trigger is set to MANUAL, Viz Mosart will not automatically play this item, but mark it as a Manual item in the UI.

> **Note: Out** and **Duration** are not yet supported for Viz Arc actions as these actions do not support such events. Only **In** time is relevant.

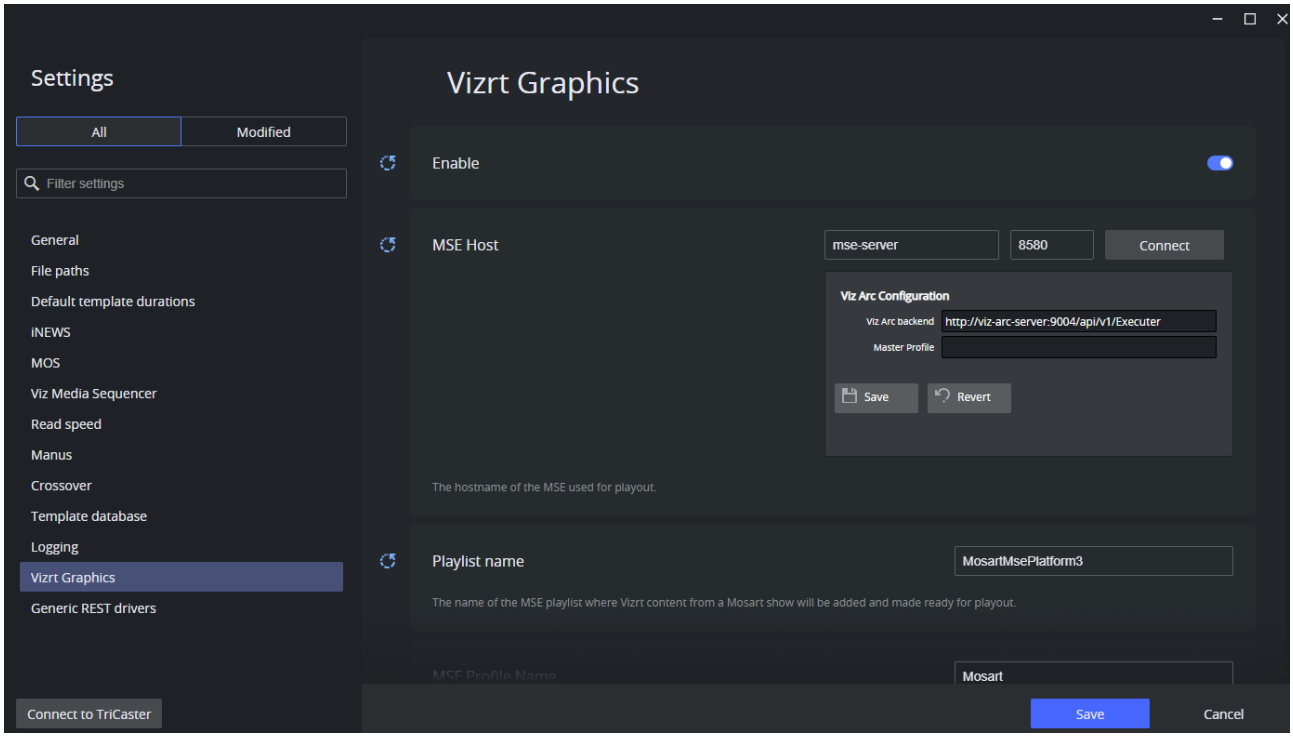
5. Use the **plus** icon in the top-right corner in Pilot Edge to drag and drop the generated MOS object into a story.



In this example, a Viz Arc action was added on a *Break* story to load a virtual set scene.

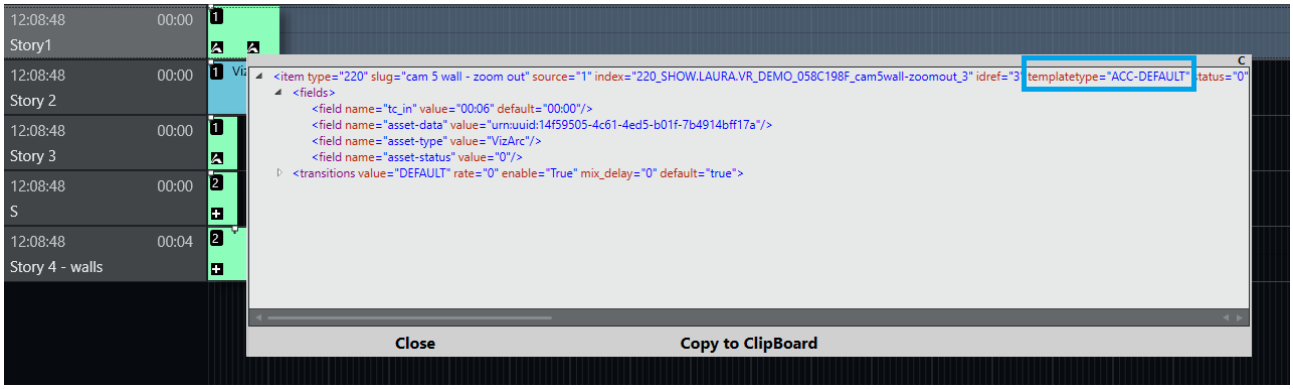
Driver Configuration

The *Vizrt Graphics* driver to communicate with Vizrt Media Sequencer through a REST API exposed by Media Sequencer needs to be configured in Manus Administrator Settings.
Refer to the Vizrt Graphics section of this guide for more details.



Viz Mosart Workflow

When a rundown containing Viz Arc MOS objects is loaded in Viz Mosart, Mosart parses and attaches the asset data to a default Accessory template:



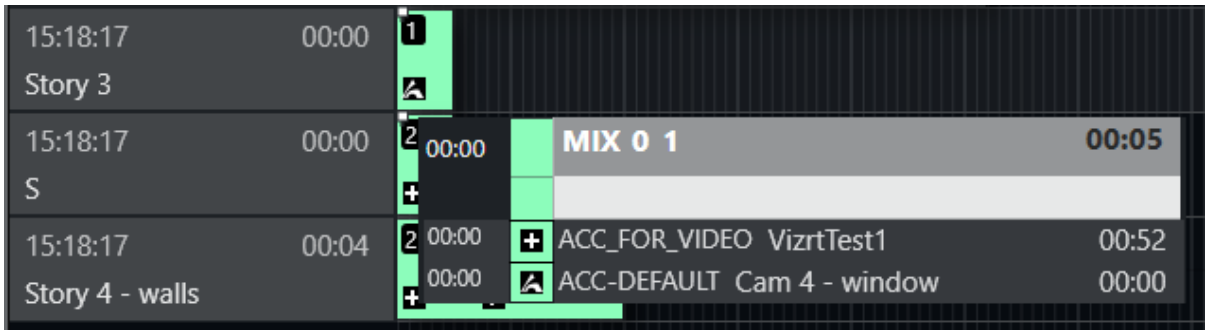
Because the Accessory item has a Viz Arc asset type attached, the visualization of the Accessory item is different to the standard *plus on a black background*.

Please refer to the *Viz Mosart User Guide* (section **Rundown Window > Story Elements > Secondary Story Elements**), for more on the visualization of Accessory items.

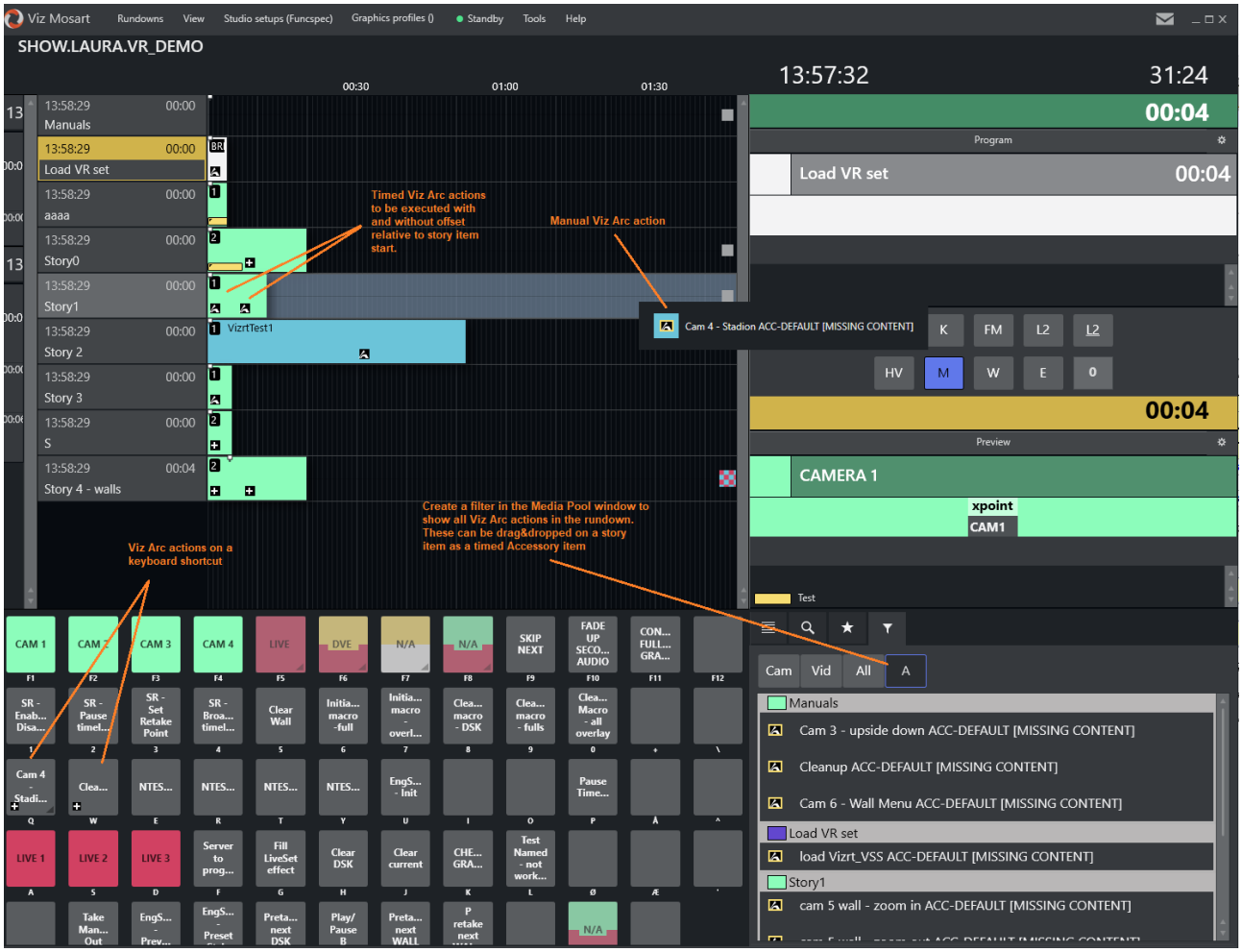
If the template does not exist in the current studio setup (template set), it is automatically generated by Mosart.

Note: Attaching Viz Arc MOS object to a custom template is *not currently* supported. If modifications are made to the default Accessory template using the Template Editor and a function is enabled within the template, that logic will be applied to *all* items utilizing that template.

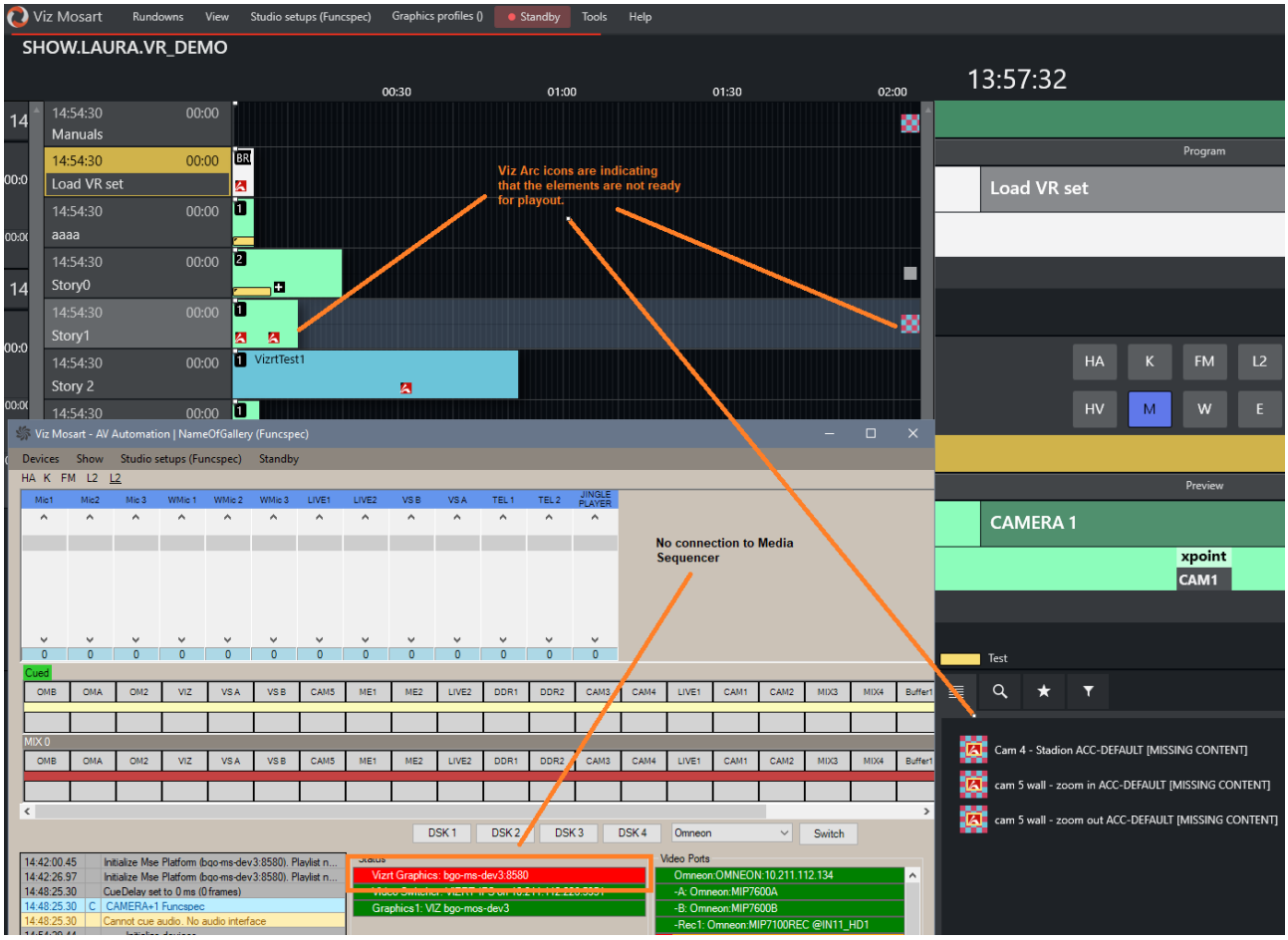
To execute other secondary actions in the same time with the Viz Arc action, a typical Accessory item can be used to execute in the same time with the Viz Arc action:



The timing and behavior (whether timed or manual) of the Accessory item are inherited from the MOS object received from the Newsroom System. Additionally, all known functionality applicable to Accessory items can also be applied to Viz Arc elements.



If the Viz Arc elements are not ready for playout because, for example, there is no connection to the Media Sequencer, this is indicated in Mosart UI with a *red* background:



You can create Viz Arc actions to

- *Initialize* the Media Sequencer (MSE) playlist when a show is loaded in Viz Mosart
- Clean up the MSE memory after each show.

These are added as manual items in the Viz Mosart rundown and activated by the operator from keyboard shortcuts.

Current Limitations

- No loading status support: A virtual set loading on the MSE may take a considerable amount of time. Currently, Viz Mosart does not show the progress of the loading status or when it is ready for playout.
- No redundancy support for the new driver: A Media Sequencer backup server can not currently be configured in case of failover.
- No Viz Arc actions supported as *control commands* on a template.

12.1.2 Viz Trio support in a Viz Mosart Show

Viz Mosart can be set up to automatically manage graphics playout from Vizrt's advanced graphics solution, *Viz Trio*.

- Creating Viz Trio Pages
- Trigger Trio with Trio Macros

Controlling Viz Trio from Viz Mosart

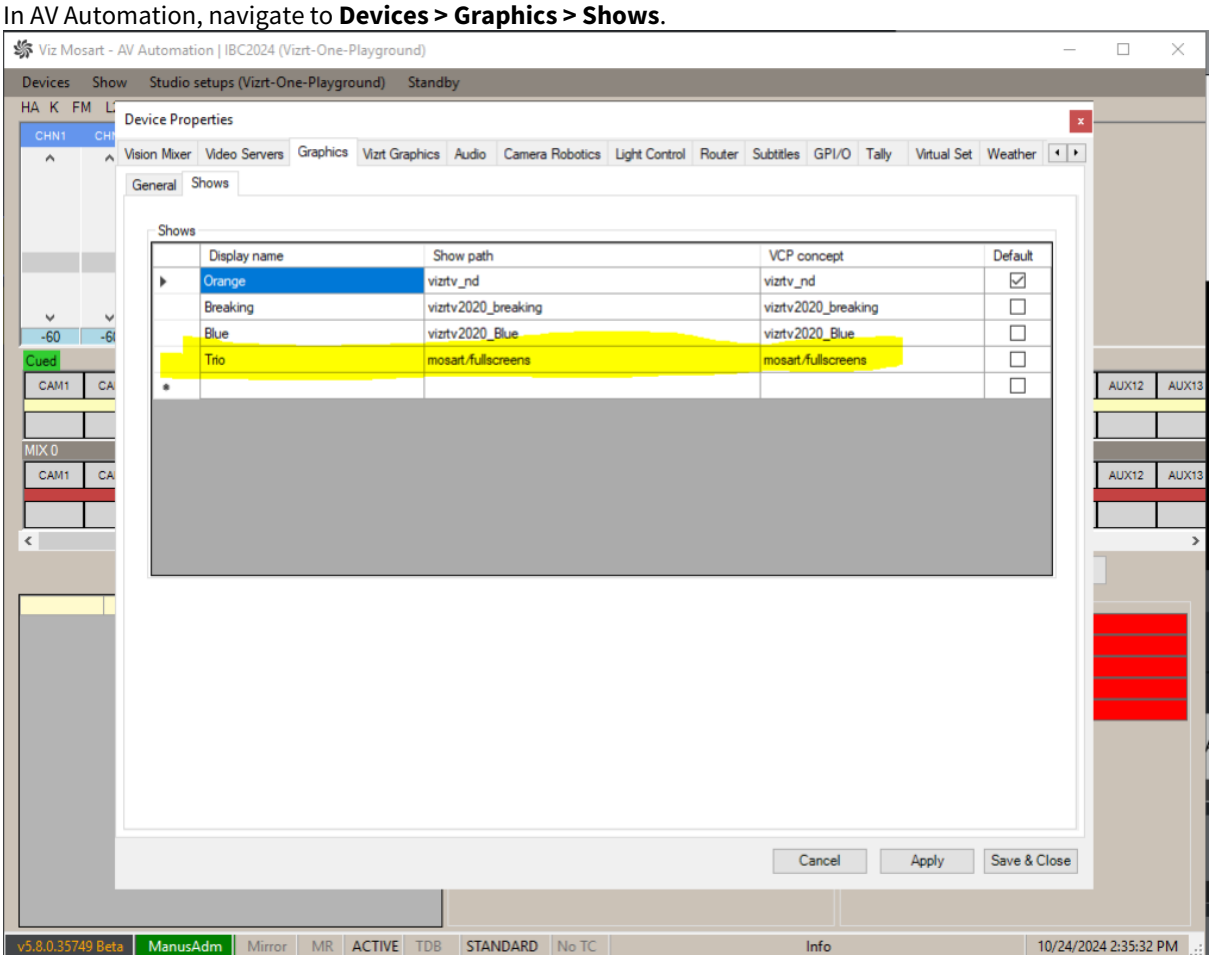
Creating Viz Trio Pages

The building block of Mosart-managed Viz Trio graphic instances, are *pages*. You create Viz Trio pages directly in Viz Mosart and then, based on input, Mosart can create pages automatically from Viz Trio templates.

This is achieved using the **Shows** configuration in AV Automation.

To create Viz Trio pages in Viz Mosart

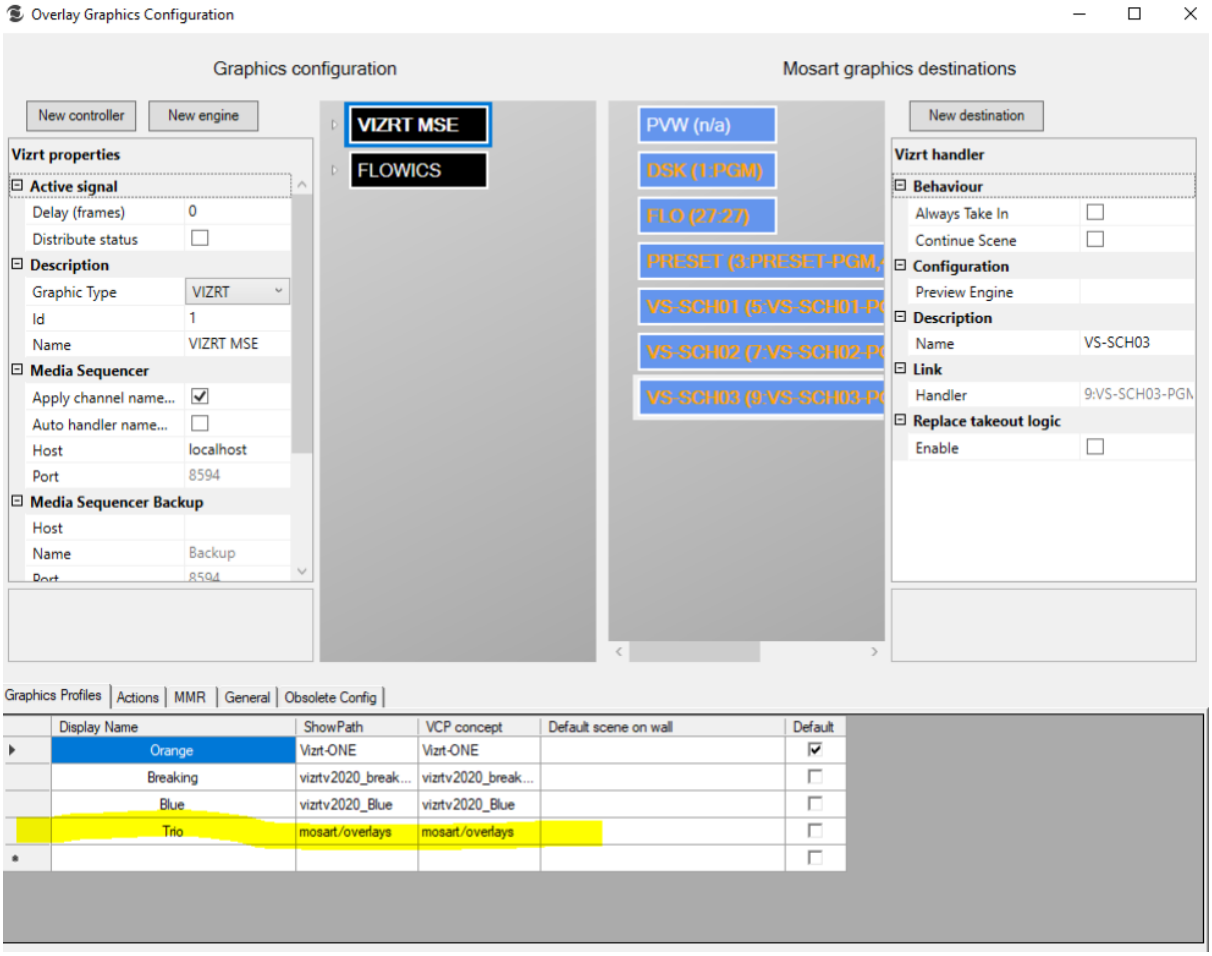
1. In AV Automation, navigate to **Devices > Graphics > Shows**.



2. Enter
   a. The name of your show in column **Display name**.
   b. The addressing of the path (found in Viz Trio) in **Show path**.

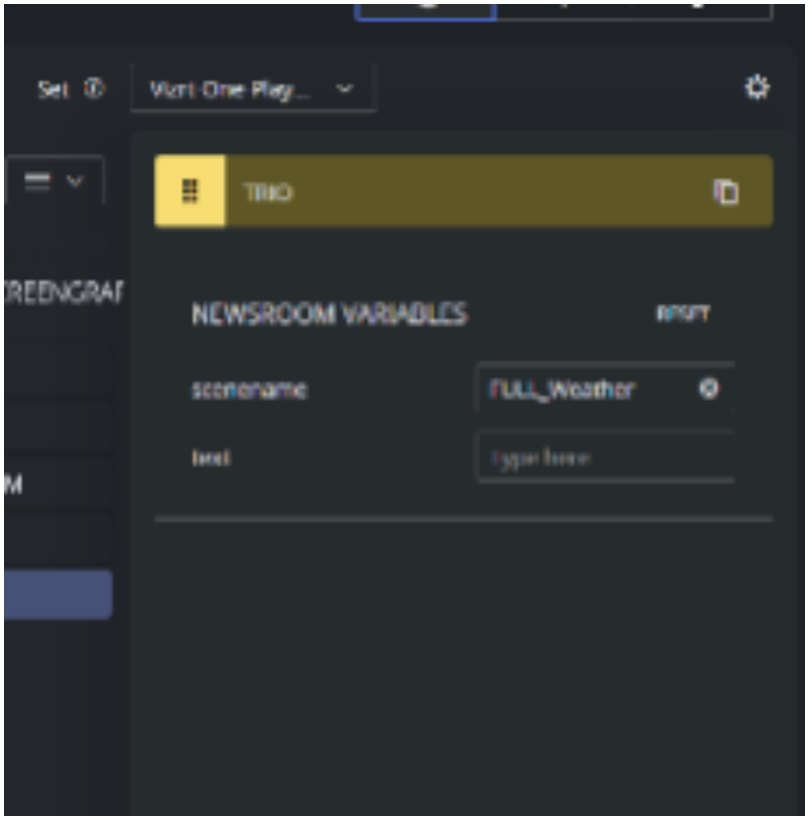
c. The Viz Content Pilot (product for creating graphics templates) concept identifier (found in Viz Trio) in **VCP concept**.

3. Open the Viz Mosart Overlay Graphics Interface from its desktop icon (or around \Program Files (x86)\Mosart Medialab\Mosart Server\MMOverLayGraphicsInterface.exe)
   a. Navigate to **Settings > Properties**.
   b. In the **Graphics configuration** panel to the left, select the Viz Media Sequencer (MSE) you are using.
   c. At the foot of the display, select the tab **Graphics Profiles**.
   d. Copy the value of *Display name* given in AV Automation above into a row in **Graphics Profiles**, as illustrated below:

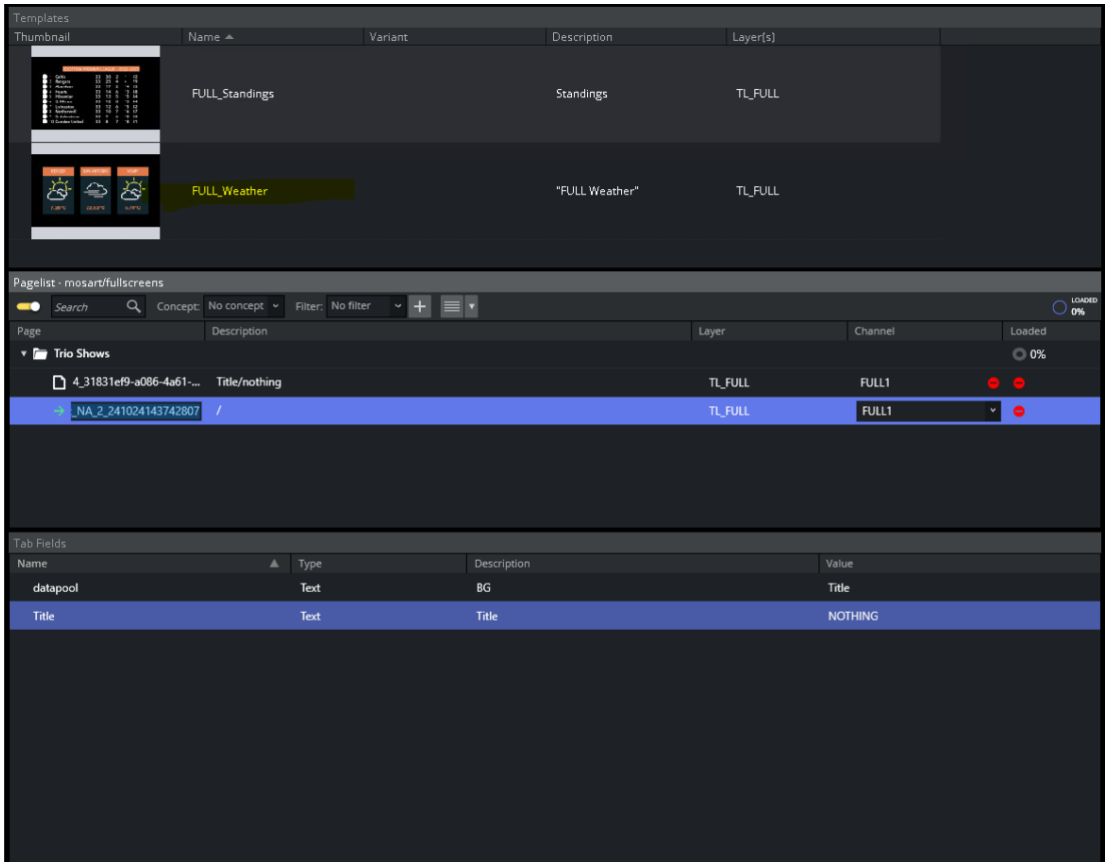


4. Back in AV Automation, from the menu bar navigate to **Devices > Template editor**.
5. Create a Fullscreen graphics (GRAPHIC) template in Template Editor using the above settings for the graphics device (where engine is the actual playout engine configured in the graphics properties):

To edit the Scene Name and Scene Text from Showmaker, rightclick and set a newsroom tag for both, but dont change the name.



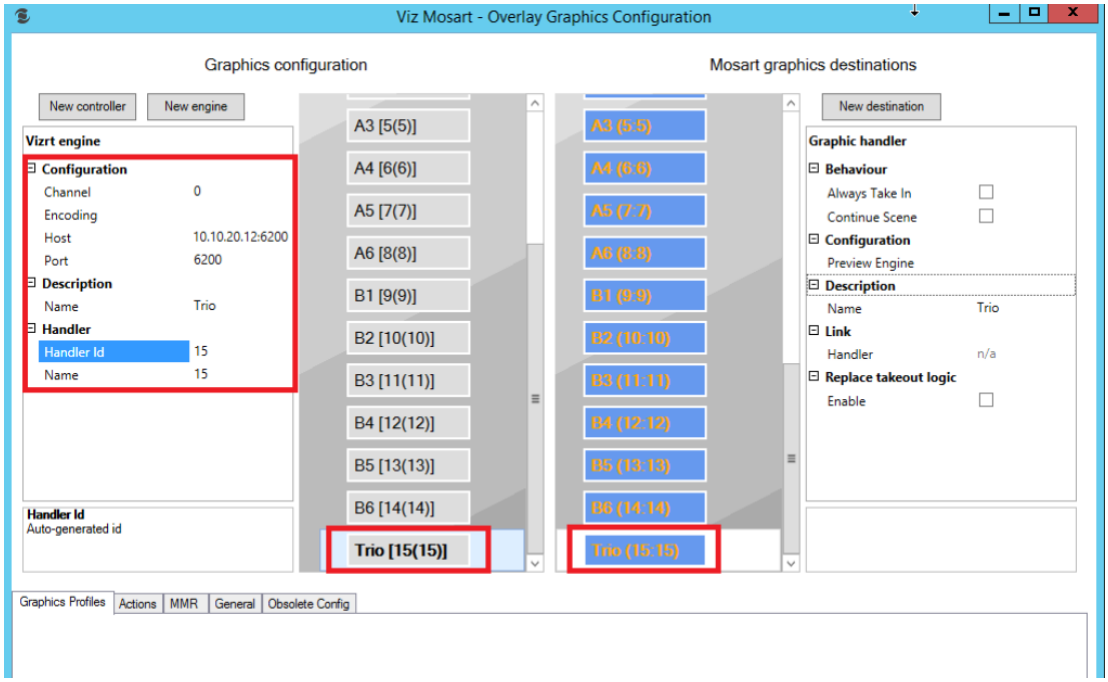
This will expose both fields in the showmaker template editor.
The scenename is the same that you find in Trio under the templates.



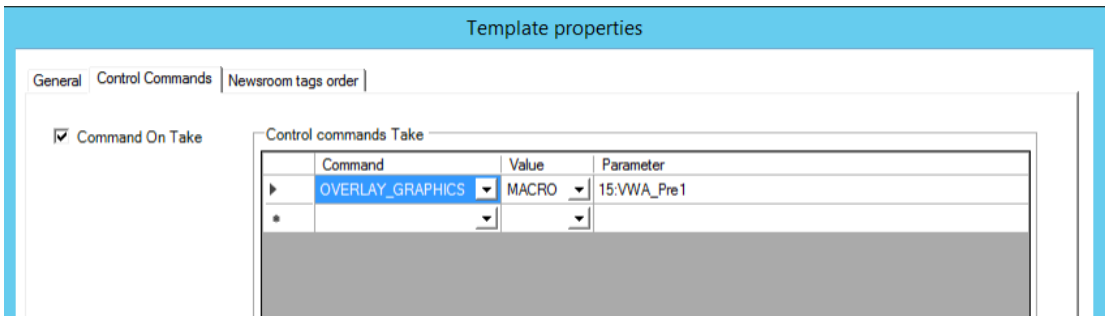
If that template has tab fields, you can enter the content in Showmaker in the text field, if you have more than one field, you can use / to separate them, like 1stline/2ndline.
Mosart will create the page and fill content in.
This also works with Overlaygraphics using the *cg workflow.

Trigger Trio with Trio Macros

Connect to the Trio Client like this



Create a template to fire a Trio macro like this:



Where 15 is the Handler ID and also part of the Parameter, VWA_Pre1 is the macro name in Trio. This could i.e take the next item or play a certain page.
Also works from keyboard shortcuts:

Alt
Shift
Ctrl
VWA Preset 1
F1
VWA Preset 1
>
Control command
Touchmode:
OnAir
None
Preview
Insert
Allow drop
Drop content only
OVERLAY_GRAPHICS
Render
Action
MACRO
Parameter
VWA_Pre1
Value
Add new command

12.1.3 Viz Engine as a Switcher

Engine Switcher Mode

With Viz Mosart in Engine Switcher mode, you can create an IP-based workflow which eliminates hardware studio switcher/mixers. In this mode:

- Viz Engine performs the function of a switcher/mixer, switching IP inputs and all rendering of DVE effects (box effects, butterflies, picture in picture, etc.) and full screen/overlay graphics.
- Engine Switcher works under exclusive control of Viz Mosart, but is *not* an integrated single unit. In Engine Switcher mode, Viz Engine and Viz Mosart run in their own environments.
  Viz Mosart templates can then control a broad scope of traditional and unique switcher features.
- On the Viz Engine side, the solution uses Viz Engine DVE based-*superchannels* for live, graphics and clip playback, instead of texture based sources.
  Superchannels can automatically swap sources in an A/B mode with animated transitions, providing further flexibility and gains in overall performance.

> Superchannels
>
> **Note**: A *superchannel* is an abstraction on top of a Viz Engine media channel. Superchannels support various asset types like images, graphics, clips, streams, live content. Each superchannel has two sub-channels, so content in each channel can be played out over an A/B workflow. One sub-channel (A) is active, while the second (B) is inactive. New content can be prepared on the inactive sub-channel, a transition effect can then run the transition from the active to inactive sub-channel, switching their roles.

- As standard, the Viz Engine processes the *dirty* feed.
  Optionally, a second Viz Engine can be set up, for a clean feed.

The following topics are presented:

- Engine Switcher Mode
- Features of the Engine Switcher
- Requirements and Limitations
- Installation
- Configuration
- Template Design
- Custom Content Transitions
- Labels
- NDI Clip Preview

> **Note**: For most recent documentation, created after this software was distributed, please visit the most recent version of this page here at the Vizrt Documentation Center.

Features of the Engine Switcher

- Preview and Program: Crosspoint can be set in preview or program (For example: cameras, external sources, clips, full screen graphics).
- Auto-trans: Instructs the switcher to perform the transition from preview to program. Viz Mosart switches to preview and sends the switcher a command to initiate the transition at a specified rate.
  This is typically achieved through a mix with rate > 0, a wipe transition, or an effect transition (using effect scenes) specified on the Viz Mosart template, in the story item provided by the NRCS or from the Viz Mosart UI.

> **Note**: An *effect* scene is a scene loaded into the Viz Engine's front layer on top of the current switcher scene. These scenes are typically used to animate on top of the actual switcher during a switch.

- Key fill crosspoint set
  The number of key buses correspond to the number of superchannels (DVE box placeholders) supported in Viz Engine.
  The switcher scene available for download contains *8* superchannels by default. You can add more superchannels, to a maximum of 16.

> **Note**:
> - While the theoretical amount of sources/superchannels can be up to 16, a high amount of sources loaded on air at the same time might lead to performance issues.
> - A key bus is represented with a superchannel. The number of superchannels is configured in VizEngineSwitcherConfig.xml.

- *Top Scene* support
  - A *top scene* is a scene loaded on to the Viz Engine front layer, on top of the current switcher scene. This type of scene is typically used for *opener* or *closer* animations for a show.
    Top scenes are called by adding a *top scene* attribute to the Viz Mosart template
  - Load (cue) and play a top scene on the Viz Engine front layer.
  - Cue a scene on-air (using a Viz Mosart *break* template).
  - Take a top scene on air.
- *Preset* support
  - The layouts for BOX effects (split screens, picture-in-picture, etc.) are stored within the scene in a *Presets* plugin. A preset contains the transformation, alpha and cropping attributes for each Viz Engine superchannel and its sub-channels.
  - Set a preset in preview by specifying the preset name and optionally providing content for the preset labels from a Viz Mosart template or from the NRCS.
  - Cue a preset on air.
  - Take a preset on air, either by hard cut or transition. This transition can be specified in either a Viz Mosart template or from the NRCS.
  - Trigger the preset style using a keyboard shortcut.
  - Go to the previous preset style using a keyboard shortcut.
- Clip support
  - Cue a clip in preview.

Copyright © 2026 Vizrt

- Cue a clip on air (using a Viz Mosart *break* template).
- Play a clip.
- Scrubbing.
- Pause (where camera and adlibs are back-to-back).
- Up to 4 clips can be cued. Using a 2x ripple, this enables an A to B transition for the single box.

Requirements and Limitations

Requirements

- Viz Engine 5.0+
- Mosart version 5.1+

Limitations

Viz Mosart 5.1 introduced Viz Engine as a switcher. The feature set is being continually expanded in successive versions. Please note the current state of the product:

- No M/Es (Mix/Effects) support other than PP (Program/Preset mixer section).
- No support for macro recall.
- No support for auxiliaries.
- No enable/disable of downstream keyers.
- No cueing of full screen graphics on air.
- No transition logic scenes.
- Overlay graphics are not a standard part of the Engine Switcher workflow. These elements must be loaded on a separate Viz Engine instance/hardware.
- Whilst both the clean and dirty feed can be enabled, the clean feed requires an optional, additional dual-channel setup.
- No main and backup support for Viz Engine from within Viz Mosart.
- Only one clip can be visible (playing in PGM). This can be achieved by either using a key fill crosspoint to play the clip within a DVE box, or via a crosspoint that presents a full screen clip. In addition,
    - Box effects with more than one clip playing *in* the box are not possible.
    - These clips and GFX scenes are always bound to the main *superchannel 1 (SC1)*.
- A preview setup has to run on separate hardware.
  This requires separate PGM and PRV Viz Engines. However, a simple clip preview is configurable.
- Viz Engine supports up to 16 superchannels. This is a theoretical maximum, operations may be limited by potential performance issues.

Installation

Sample Scenes

In addition to a standard Viz Mosart setup, Engine Switcher builds on Viz Engine *scenes* for template-driven operations in Viz Mosart.
To simplify the show design process, sample scenes are provided by Vizrt.

To obtain sample Engine Switcher scenes

1. Authenticate to Vizrt's FTP at https://download.vizrt.com/
2. Navigate to products/VizMosart/Engine Switcher Scenes
3. Download locally, then import the switcher scene to Viz Graphic Hub
4. Set the path to the Switcher Scene in VizEngineSwitcherConfig.xml

Note: For latest installer files and template samples, always refer to the latest version of this documentation at the Vizrt Documentation Center.

Configuration

For first-time configuration, all the setup steps below must be followed:

1. AV Setup with Cross Point Naming and Mapping.
2. Video Switcher Setup
3. Graphics
4. Video Server and Port Configurations
5. Manual Configuration of the Vizrt Media Sequencer Engine (MSE)
   and then to perform
6. Template Design.

AV Setup with Cross Point Naming and Mapping

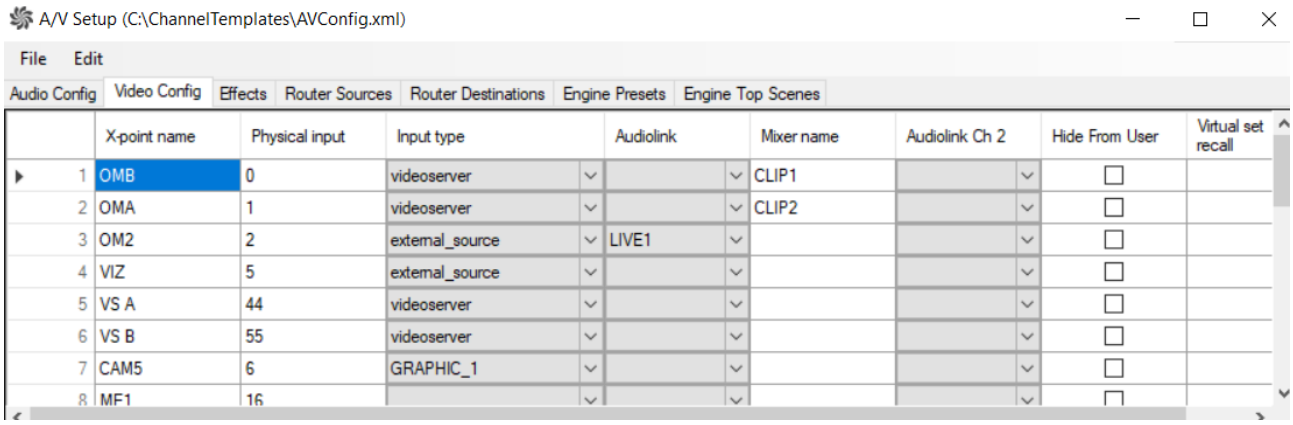
When Viz Mosart instructs a standard video switcher to put something in preview or on air, it is sufficient to configure AV Config file with a list of crosspoints that correspond to the video signal input configurations defined on the switcher.

In Engine Switcher mode, employing a Viz Engine as a video switcher, these standard rules no longer apply. There are no video signal inputs for the video server ports in a Viz Engine.
Instead, the crosspoints on the video server ports in AV Automation must be configured to point directly to the port names configured on the Viz Engine.

Defining Sources

- Source ID from 1 - N.
- CLIP X-points from 1-6 are mandatory and used for internal switching.
- GFX X-points are mandatory for graphics.

Tip: The X-point name can either be
- the source ID as defined on the Switcher side
  or
- a *Viz Mosart user-friendly name*.
  In this case, then *source ID* (as defined on the Switcher side) should be entered in the **Mixer name** column.

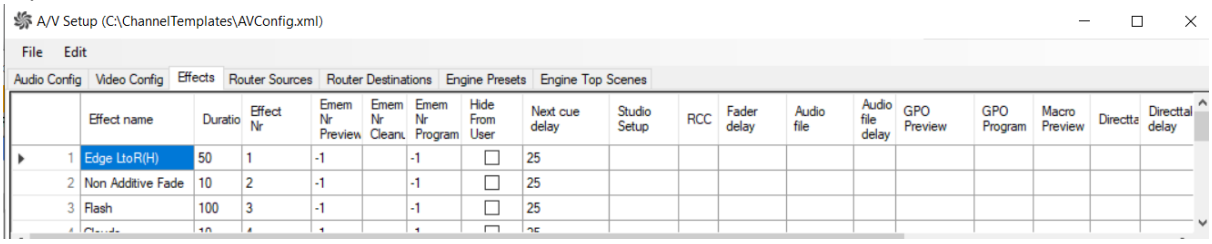


	X-point name	Physical input	Input type	Audiolink	Mixer name	Audiolink Ch 2	Hide From User	Virtual set recall
1	OMB	0	videoserver		CLIP1		☐	
2	OMA	1	videoserver		CLIP2		☐	
3	OM2	2	external_source	LIVE1			☐	
4	VIZ	5	external_source				☐	
5	VS A	44	videoserver				☐	
6	VS B	55	videoserver				☐	
7	CAM5	6	GRAPHIC_1				☐	
8	MF1	16					☐	

By configuring these crosspoints on the video server ports in AV Automation or referring to them in Viz Mosart templates, these definitions enable standard Viz Mosart clip execution.

Effects

- Effects need a unique *Emem Nr* for calling a specific effect scene.
- Scenes are always prefixed with the string *effect_x* in the designated Graphic Hub effect Scene folder. x represents the Emen Nr defined in the configuration.
- Effect IDs must not be higher than *80*, since numbers 80+ are reserved for transition effects within superchannels.

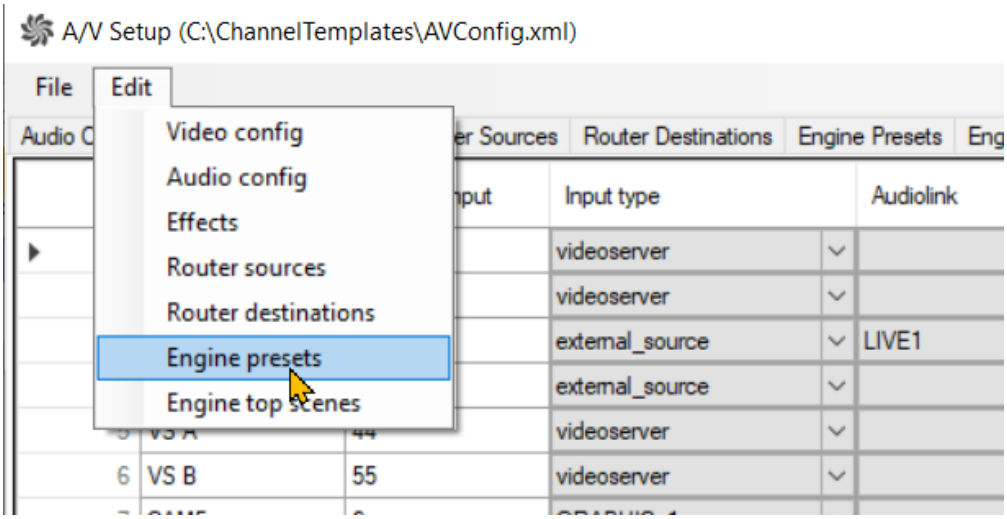


	Effect name	Duratio	Effect Nr	Emem Nr Preview	Emem Nr Cleanu	Emem Nr Program	Hide From User	Next cue delay	Studio Setup	RCC	Fader delay	Audio file	Audio file delay	GPO Preview	GPO Program	Macro Preview	Directta	Directtal delay
1	Edge LtoR(H)	50	1	-1		-1	☐	25										
2	Non Additive Fade	10	2	-1		-1	☐	25										
3	Flash	100	3	-1		-1	☐	25										

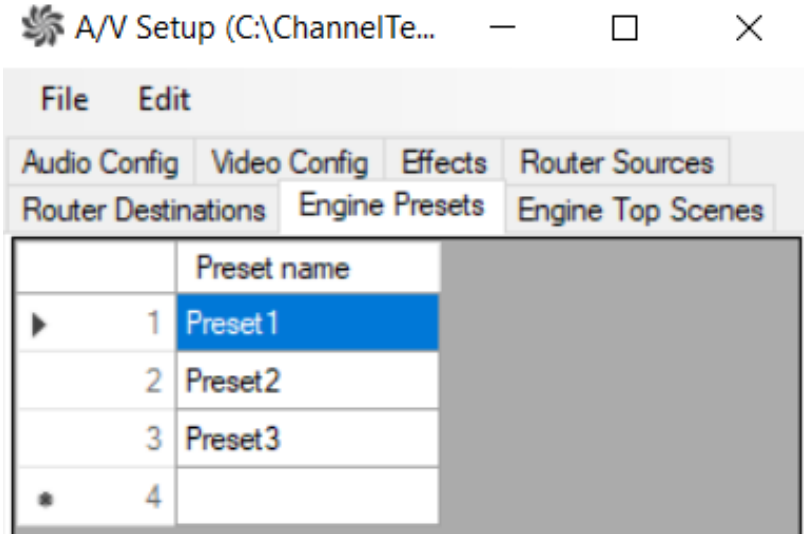
Presets

> **Tip**: The concept *Presets*, is introduced below.

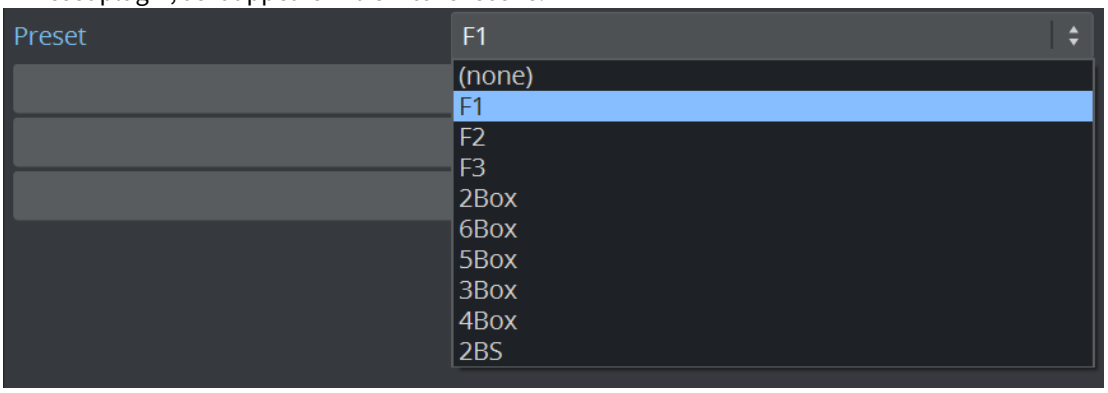
- In Viz Mosart, you see the preset names in the A/V Setup (`AvConfig.xm` l) file.



- Preset names correspond directly to the presets defined in the switcher scene:



- A Preset plugin, as it appears in a switcher scene:



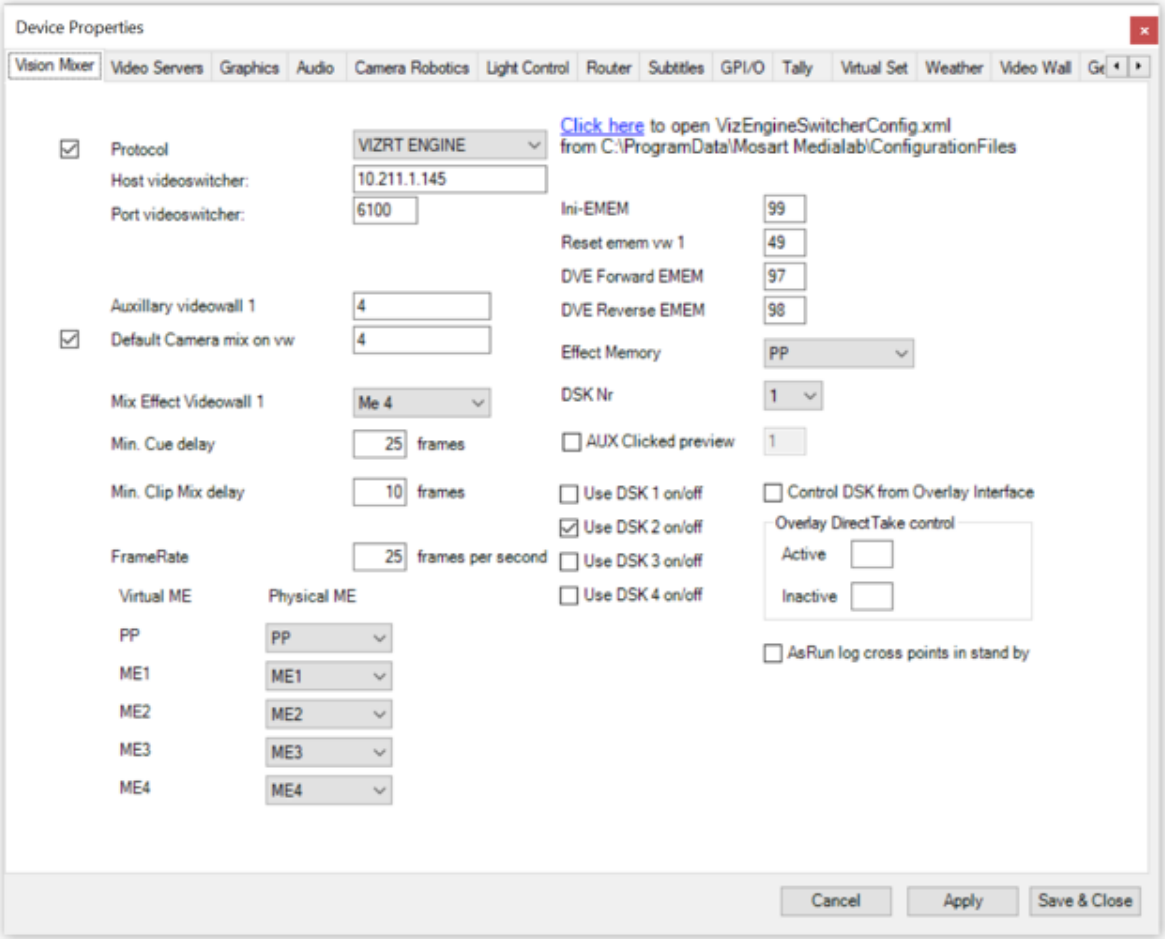
Info: F1, F2 etc. are the default full screen options for each superchannel. F1 is *mandatory*.

Video Switcher Setup

Video switcher setup is performed in Viz Mosart.

In a Viz Engine Switcher setup, there are no MEs, just PP.

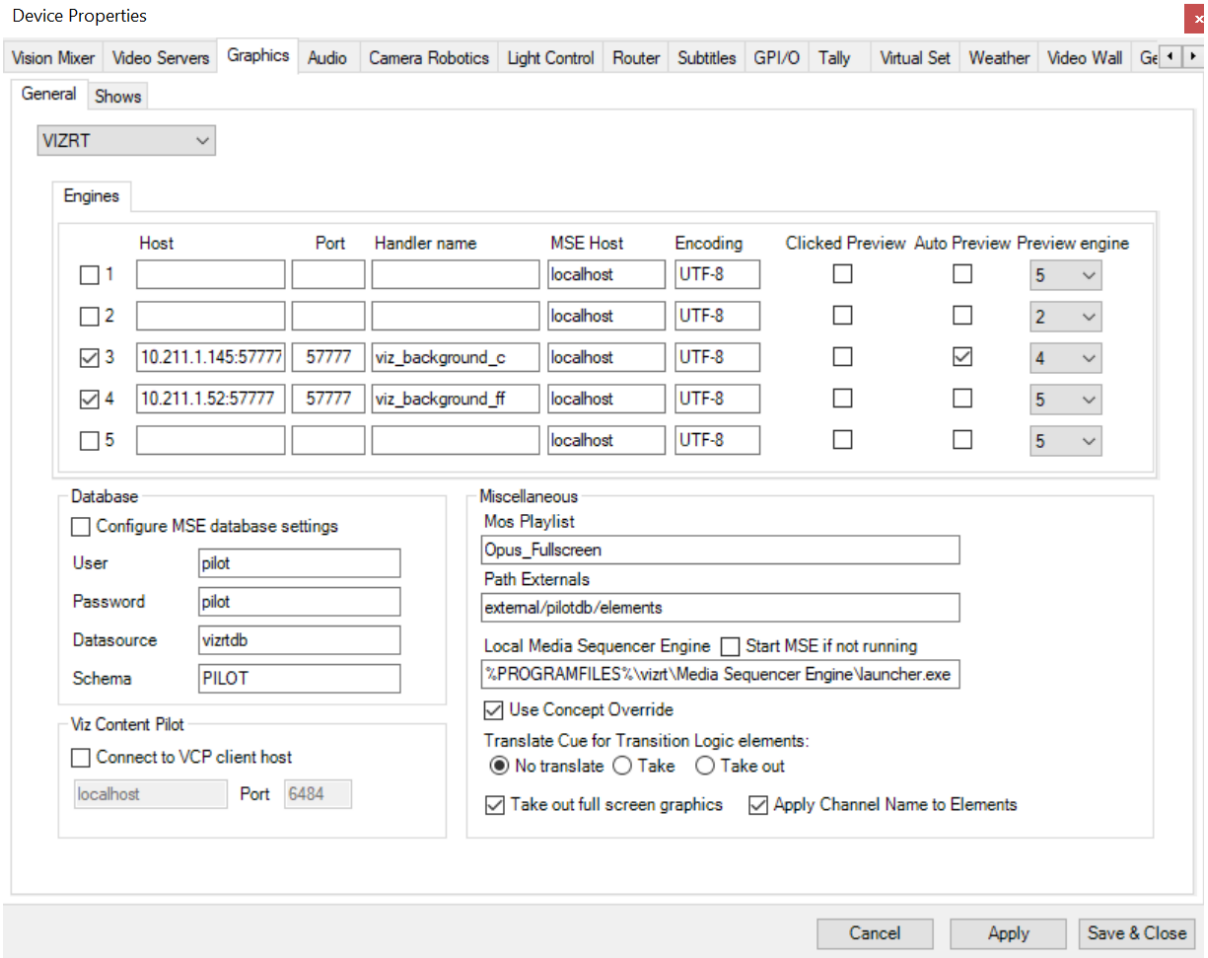
1. Navigate to **AV Automation > Devices > Properties > Vision Mixer** tab.
   From the **Protocol** drop-down menu, select *VIZRT ENGINE*.



2. In addition to the default values, enter the following:
   a. **Host videoswitcher**: Location of the Viz Engine.
   b. **Port videoswitcher**: default `6100`.
   c. Tick the box to the left of **Protocol**. This indicates that you want to control the vision switcher (mixer).
3. Click **Save & Close**.
4. From the *Mosart Installation folder > Mosart Server > Configuration Files* copy VizEngineSwitcherConfig.xml file into your Mosart working folder (*C:\channeltemplates* or *C:\ProgramData\Mosart Medialab\ConfigurationFiles*) and configure:
   a. **ControlScenePat**: The path to the Switcher scene located in the Viz Engine.
   b. **NumberOfKeyBuses**: The number of superchannels defined in the Viz Engine.

Graphics

1. Select the **Graphics** tab.

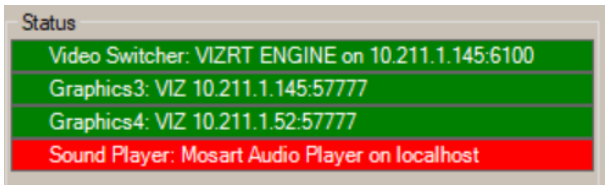


2. On the displayed **General** tab, in the field **Host**, add the Viz Engine entry for PGM.

> **Note**: A second Viz Engine, running on separate hardware, can be added for optional PRV.
> - Enable the checkbox **Auto Preview** option and link it to the PGM engine. See Setup Second Engine.

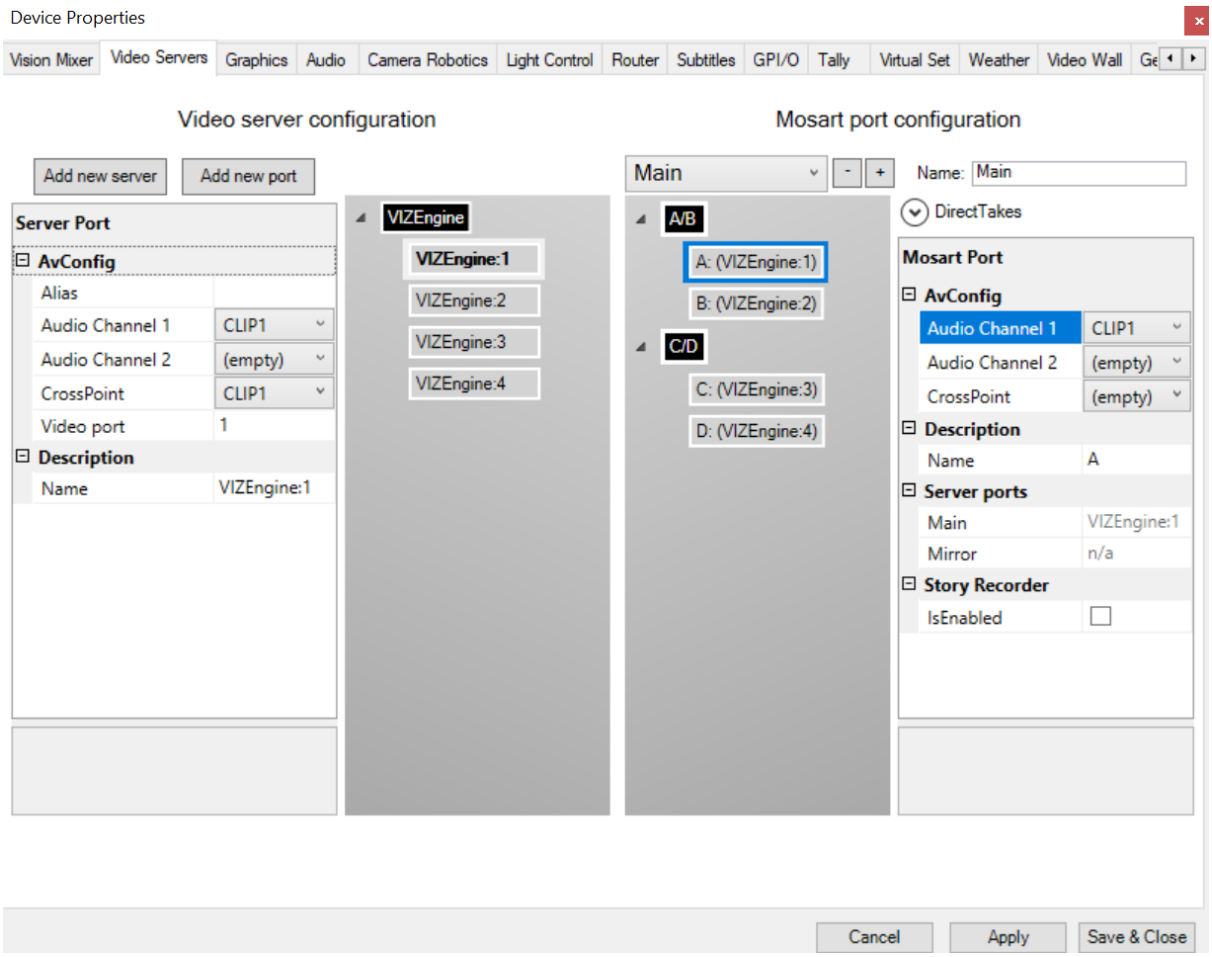
3. Add the designated *superchannel* port for each Viz Engine (default is *57777*).
4. Tick the **Auto Preview** box for the PGM Viz Engine.
5. From the dropdown box under the **Preview engine** column, assign the identity of the PRV Viz Engine to it.
6. Follow the other guiding values as illustrated above, then click **Save & Close**.

These show up in AV Automation:

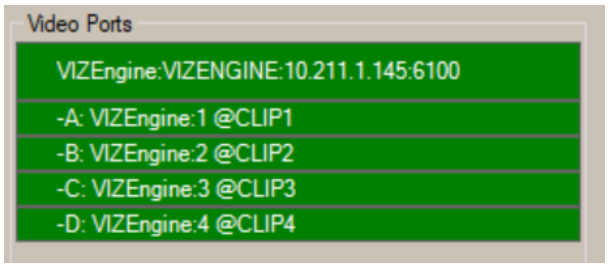


Video Server and Port Configurations

- Select the **Video Servers** tab and add four server ports to create two clip server pairs:
  - A/B
  - C/D



These show up in AV Automation:



Viz Engine Configuration

The following setup is performed on the Viz Engine.

Standard Viz Engine configurations

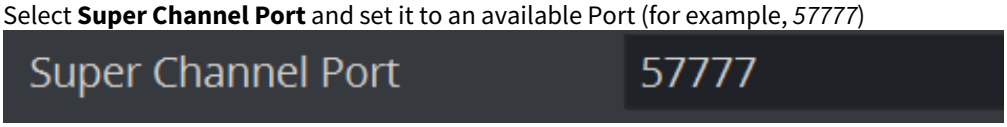
You will need to provide basic Viz Engine configurations that include definitions of:

- Graphic Hub connection
- Output format
- Licenses
- Amount of inputs.

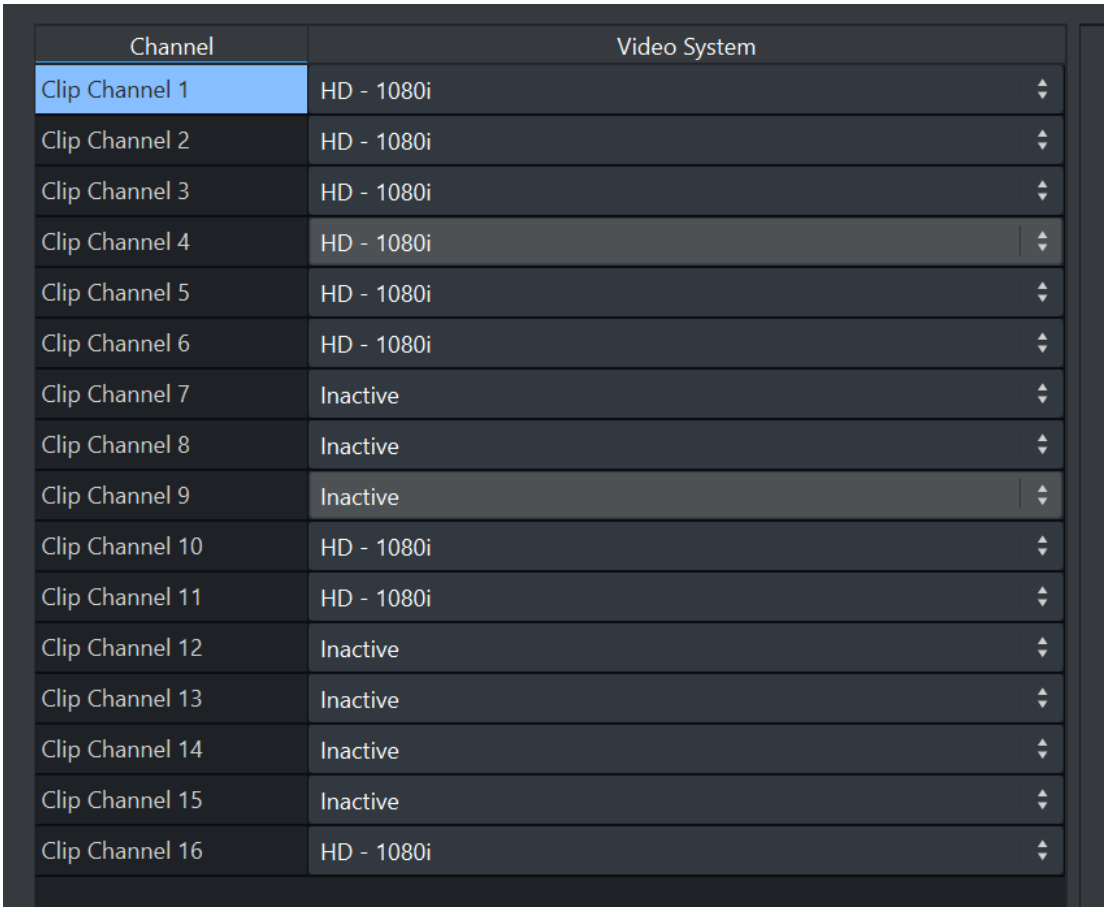
If these are not already in place, refer to the *Viz Engine Administrator Guide*.

Additional Viz Engine configurations for an Engine Switcher setup

1. Navigate to the configuration section **Communication**.
2. Select **Super Channel Port** and set it to an available Port (for example, *57777*)



3. Clip channels: Activate the following clip channels:



Channel	Video System
Clip Channel 1	HD - 1080i
Clip Channel 2	HD - 1080i
Clip Channel 3	HD - 1080i
Clip Channel 4	HD - 1080i
Clip Channel 5	HD - 1080i
Clip Channel 6	HD - 1080i
Clip Channel 7	Inactive
Clip Channel 8	Inactive
Clip Channel 9	Inactive
Clip Channel 10	HD - 1080i
Clip Channel 11	HD - 1080i
Clip Channel 12	Inactive
Clip Channel 13	Inactive
Clip Channel 14	Inactive
Clip Channel 15	Inactive
Clip Channel 16	HD - 1080i

- Clip channels *1-4* and *10-11* are for *full screen* and *box* clip playback.
- Clip channels *5-6* are optional for *videowall* integration playback (not included in default setup).
- Clip channel *16* is for clips loaded in *top* scenes.

Switcher Scene Configuration

The Switcher scene comes with a separate configuration file.

- Copy this file to C:\ProgramData\vizrt\VizEngine.

This file enables modification of switching behavior, delays and the adding of a separate Preview engine.

Manual Configuration of the Vizrt Media Sequencer Engine (MSE)

The MSE handlers that you use must be set to *superchannel* mode.

This is performed manually in the VDOM configuration file.

1. Navigate to *http://localhost:8580/app/vdomconfig/vdomconfig.html*, node */scheduler/<handler name>*. For example, in the AV Automation screenshot above of the **Graphics** tab, the handler name is *viz_background_c.*
2. Add the following entries to both handlers:

```
<entry name="superchannel"/>
```

```
<ignore_cue_operation>no</ignore_cue_operation>
<ignore_out_operation>yes</ignore_out_operation>
```

Example of the MSE VDOM File

```
<handler connected="yes" mode="enabled" status="good: Commands processed" type="viz"
 name="viz_background_c">
    <entry name="encoding">UTF-8</entry>
    <entry name="state">
        <entry name="last">
            <entry name="element">/external/pilotdb/elements/406</entry>
        </entry>
    </entry>
    <entry name="renderer">
        <entry name="10.211.1.145:57777">
            <entry name="active"/>
        </entry>
    </entry>
    <superchannel/>
    <entry name="publishing_point_uri"/>
    <entry name="publishing_point_atom_id"/>
    <entry name="info"/>
    <entry name="superchannel"/>
    <ignore_cue_operation>no</ignore_cue_operation>
    <ignore_out_operation>yes</ignore_out_operation>
    <command_timeout>30.</command_timeout>
</handler>
```

Template Design

A show with video switching performed by Engine Switcher, is controlled exactly like any standard video switcher-driven show, from a rundown with associated Viz Mosart templates.

Tip:

- For an overview of template design with examples, please refer to the section Template Examples, section Switcher Plugin.
- To download sample scenes, see To obtain sample Engine Switcher scene.

Introduction to Presets, Top scenes and Effects

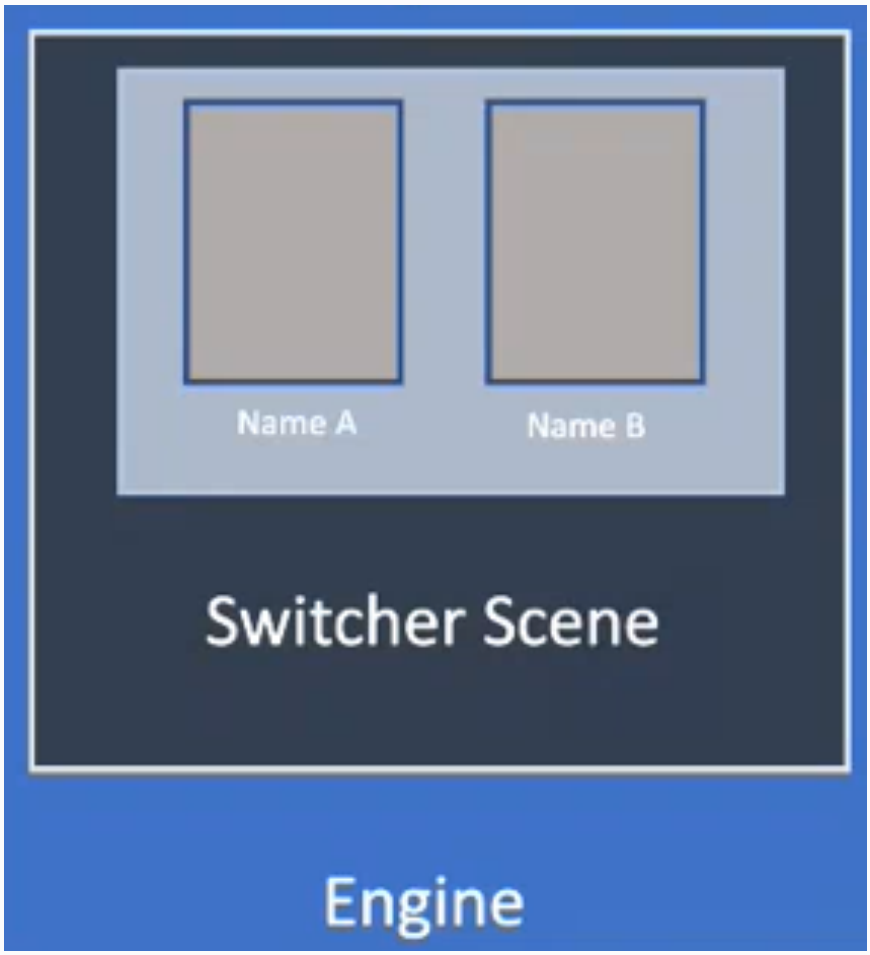
Viz Engine Switcher operations depend on distinct criteria, controlled from a Viz Mosart template. All settings are made during template design, as described in the section Template Editor.
Three concepts for Engine Switcher operations are:

- Presets
- Top Scenes
- Effect Scenes.

Presets

Info: The term *preset* originates from scene design in the Viz Engine. The composition of the scene is defined by x,y positions of a collection of 'windows' (two in the illustration below) in a Viz Engine *superchannel*. These definitions describe:

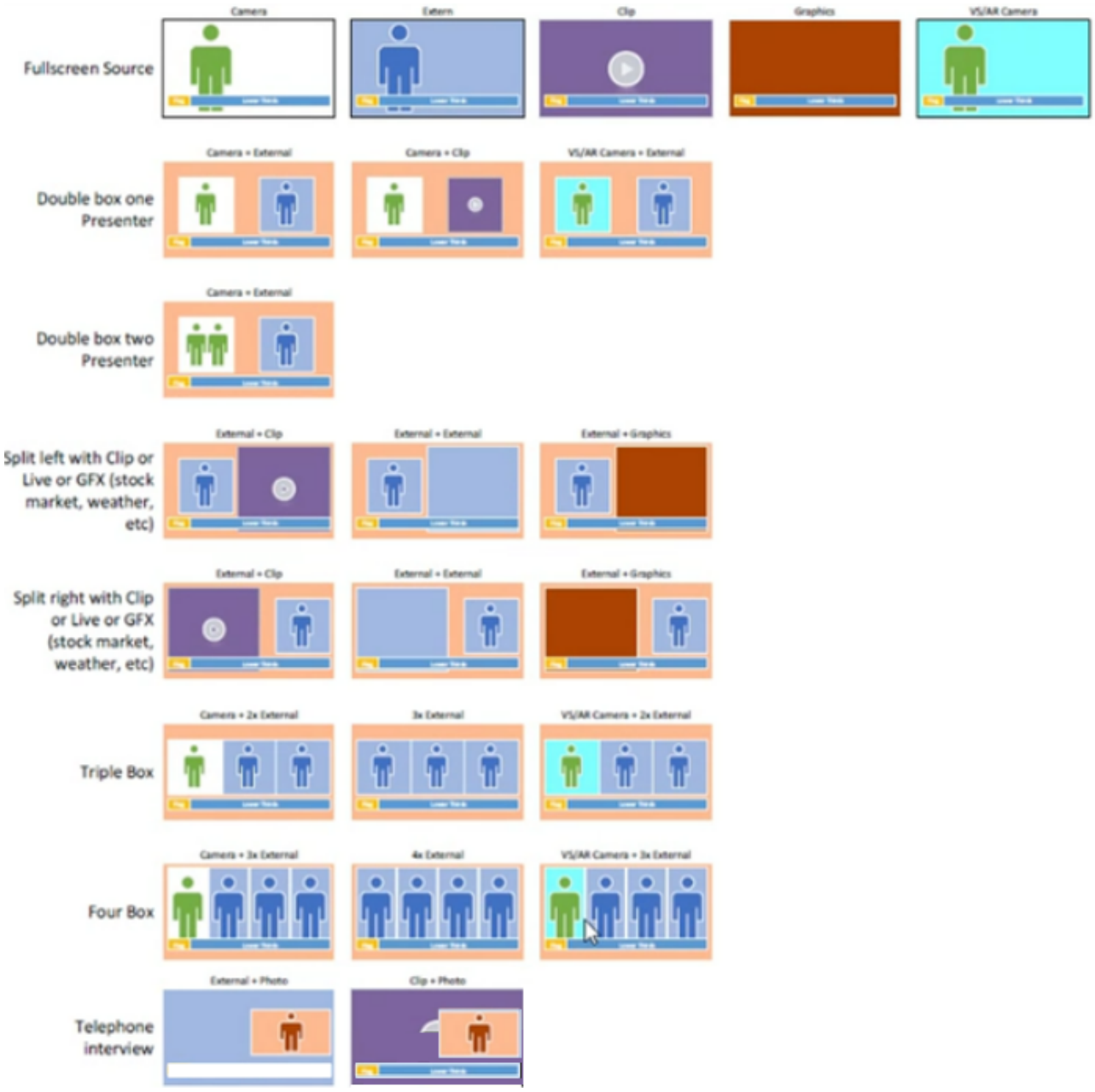
- The *number* of windows
- The p*osition* of the windows
- The size of each window.



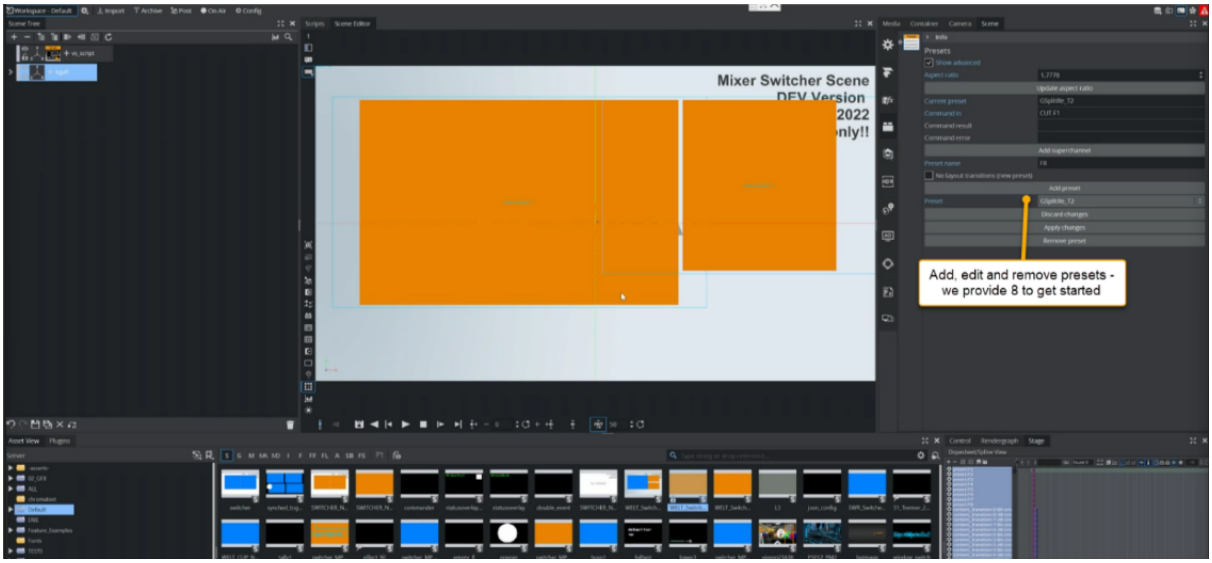
Each composition, saved on the Viz Engine, is called a *preset*.

Note: Superchannel 1 is considered the main superchannel for full screen program switches. The preset *F1* is mandatory for this workflow and must *not* be altered.

Viz Mosart includes a sample Switcher scene with eight presets.

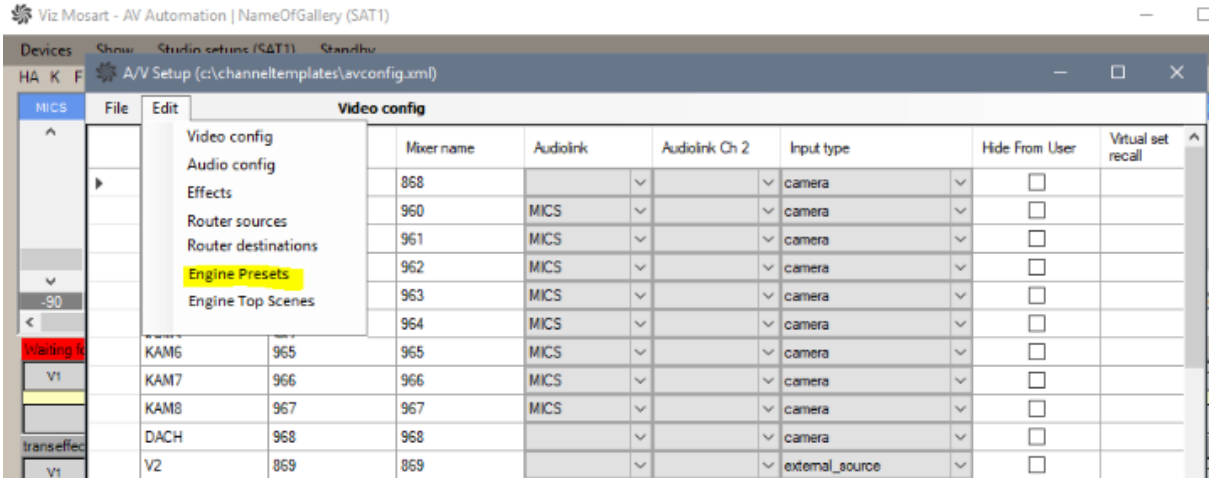


- Use Viz Artist to edit any preset.



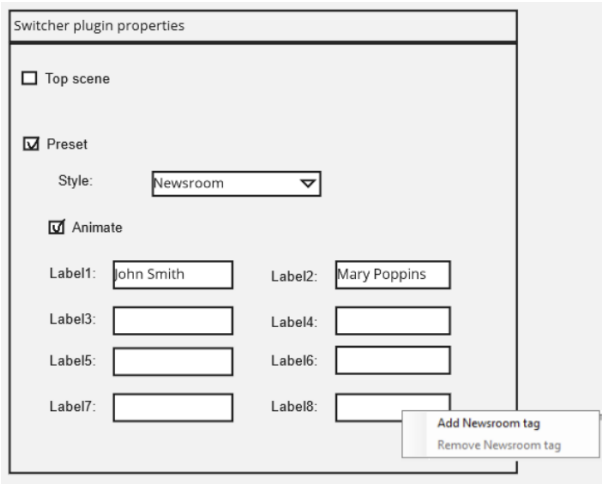
For more details, refer to the *Viz Artist User Guide*.

- In Viz Mosart, you select the Preset name from the list of preset names that have been configured in A/V Setup (AvConfig.xml) file.



- Transition style
  The transition *style* (hard-cut or animation), and several *label* values can be specified.
  For example, in a DVE template (where **Switcher plugin** for the device is available), you can select the layout and look of the DVE effect by specifying a preset style (**Style**), the preset transition and the DVE window

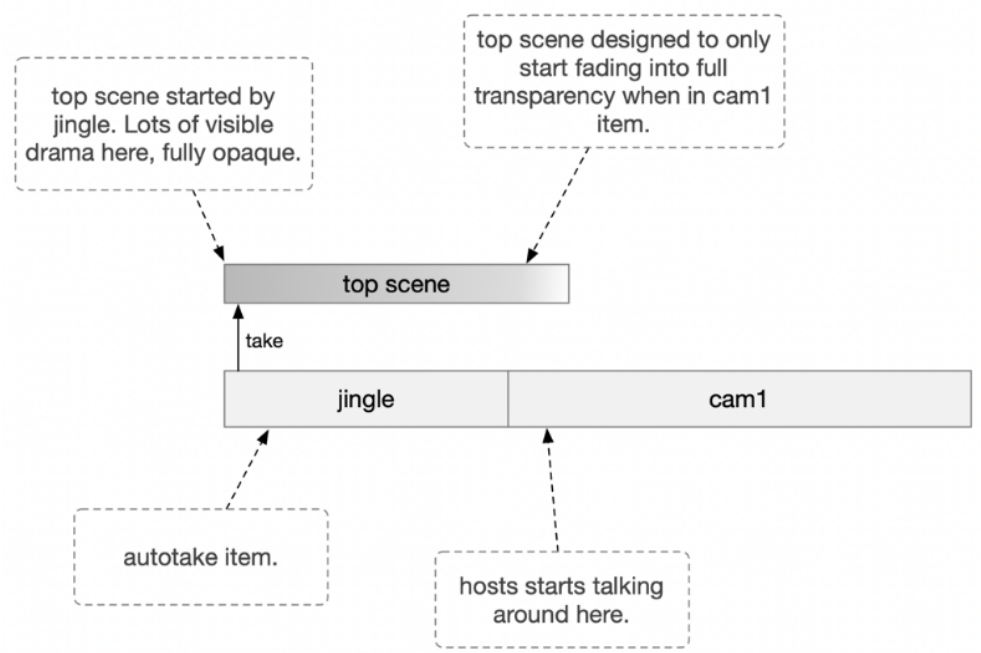
labels (**Label1** - **Label8**), as required:



You can also provide static values directly on the template. Additionally, newsroom tags can be configured to enable control from the NRCS.

Top Scenes

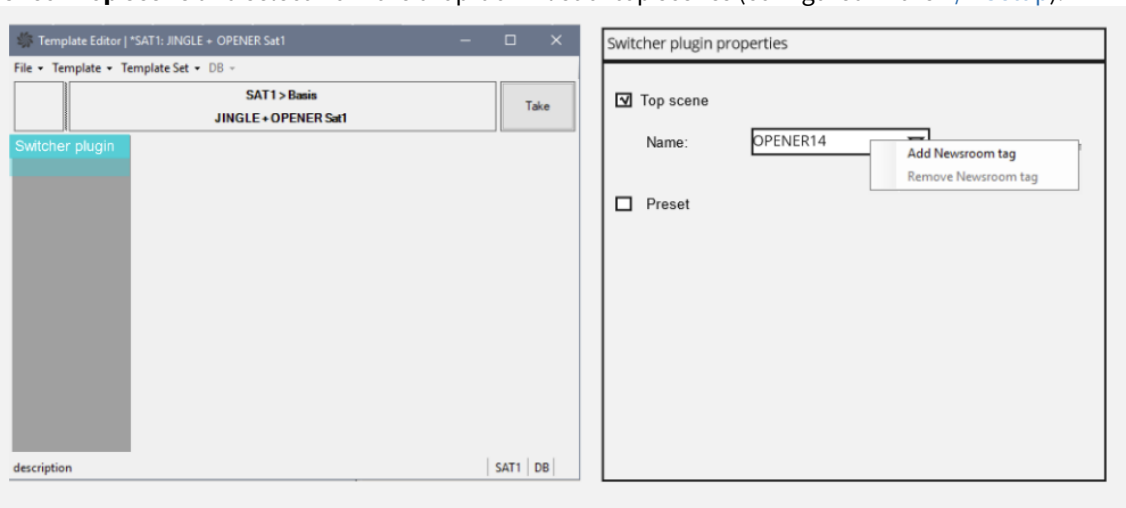
A typical use of top scenes is as an *opener*. For example, the top scene is played to introduce the Eight O'clock News show with many visual effects that fade to the following camera scene, when the news host takes over.



To Create a template to play a top scene

1. Select an appropriate template, for example *Graphic* or *Jingle*, corresponding to your Engine Switcher-controlled operation.

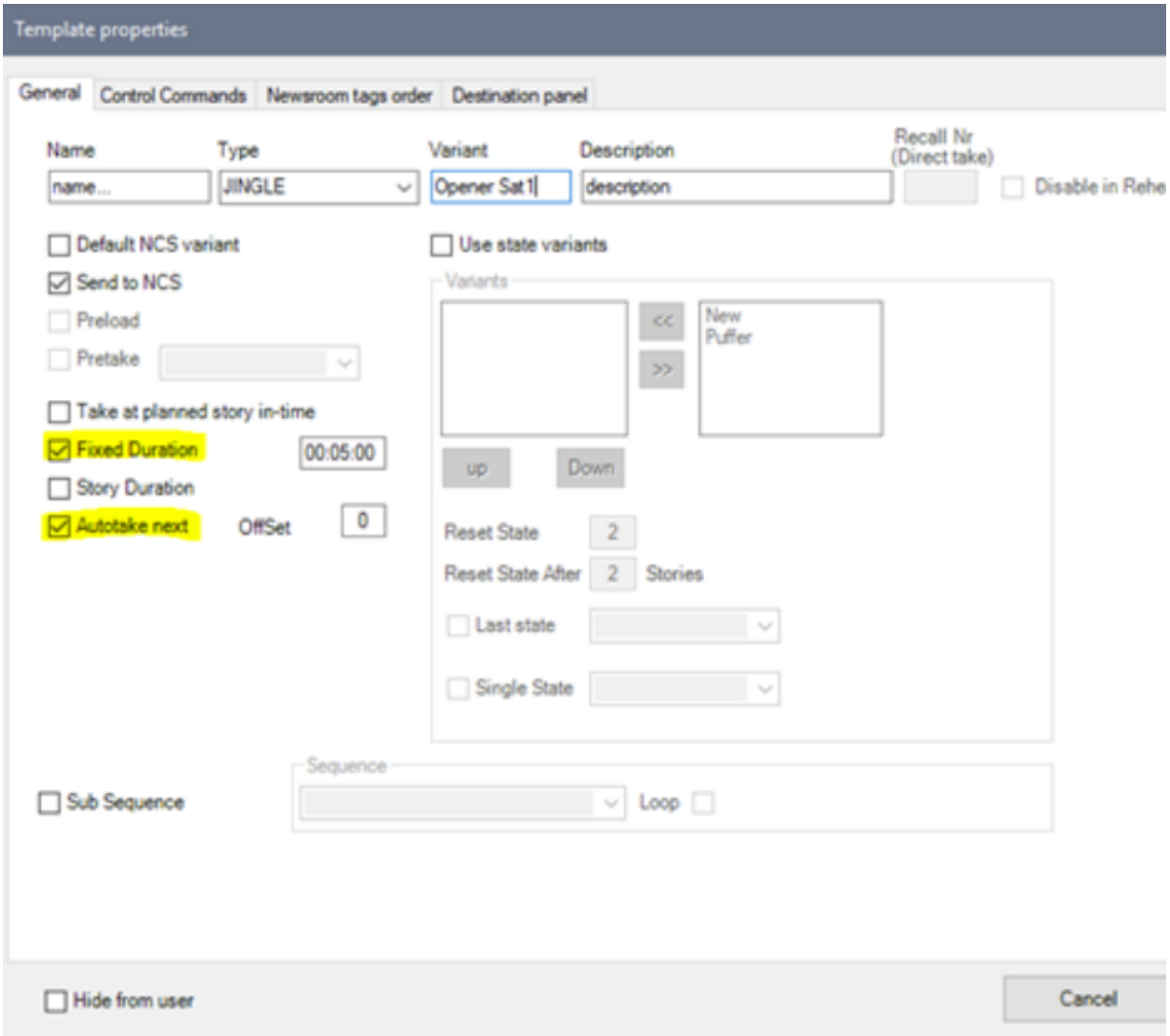
2. Enable **Switcher plugin** for the device.
3. From the **Switcher plugin properties** menu, specify a top scene name:
    a. None (**Top scene** unchecked)
    b. Check **Top scene** and select from the drop-down list of top scenes (configured in the A/V Setup).



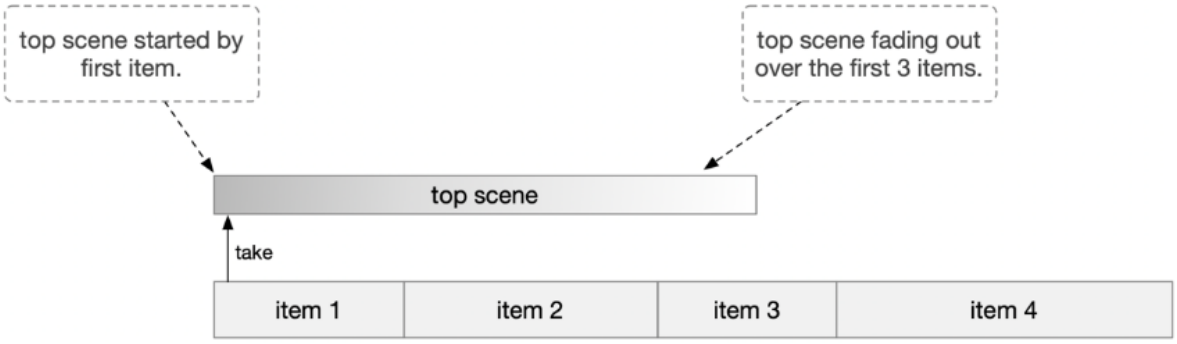
Time-related configurations

There are two time-related aspects for top scenes (similar to handling clips):

1. The time it takes for the scene to *load*. This is controlled automatically by Viz Mosart when cueing the story item.
2. The time it takes for the scene to *run*. This is controlled with the template configurations **Fixed Duration** and **Autotake next**.



- Top scenes are started from within a story item and if long enough, continue to play even when the item ends.
- Top scenes ignore story items boundaries. The take out of a story item must match the animation length of the top scene.
- Top scenes can be played on top of anything else.



To create a template that plays a **Viz Engine Top scene**

The sample script below (code block) performs this operation:

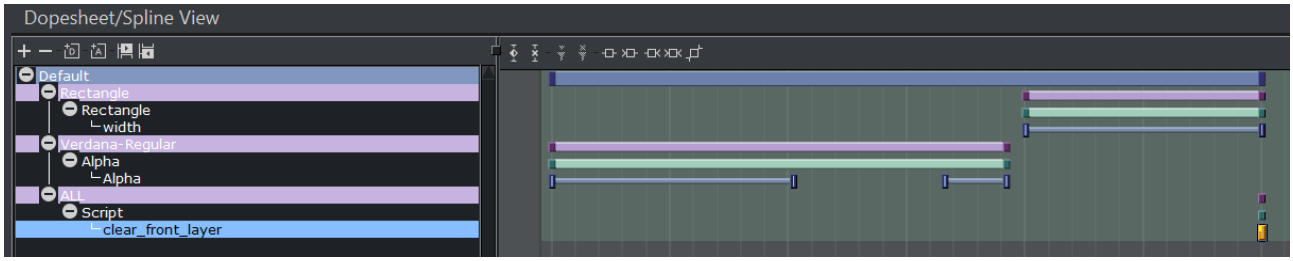
1. Initially the scene is fully opaque, covering the switcher completely.
2. On *Take,* an animation starts.
3. After a pre-defined time, the openers content disappears (animates or fades out).

The Engine Switcher workflow does not need to contain a script; it is purposefully designed for simplicity. A script *can* be added, to call the "clear_front_layer" routine of the switcher script which will force-remove the top scene from the front layer after the animation is finished.

Example Script

```
'-----------------------------------------------------------------------------------------
-------------------------------------
sub OnInitParameters()
    RegisterPushButton("clear_front_layer", "Opener Close", 1)
end sub
'-----------------------------------------------------------------------------------------
-------------------------------------
sub OnExecAction(buttonId As Integer)
    if buttonid = 1 then clear_front_layer()
end Sub
'-----------------------------------------------------------------------------------------
-------------------------------------
sub clear_front_layer()
    dim commandStr as string = "RENDERER*MAIN_LAYER*SCRIPT INVOKE clear_front_layer"
    println 5, "--------------- TOP SCENE OUT ----------------"
    System.SendRemoteCommand("localhost", 6100, commandStr, 0)
end sub
'-----------------------------------------------------------------------------------------
-------------------------------------
```

In this case a **clear_front_layer** button is added to the animation in the Stage



Effect Scenes

Effect scenes, similar to Top scenes, are loaded to the front layer of the Engine Switcher for an upcoming switch between sources. These scenes are loaded automatically once a transition effect is triggered in Viz Mosart. Scene names must be added to AV configuration and are typically named "effect_<id>".

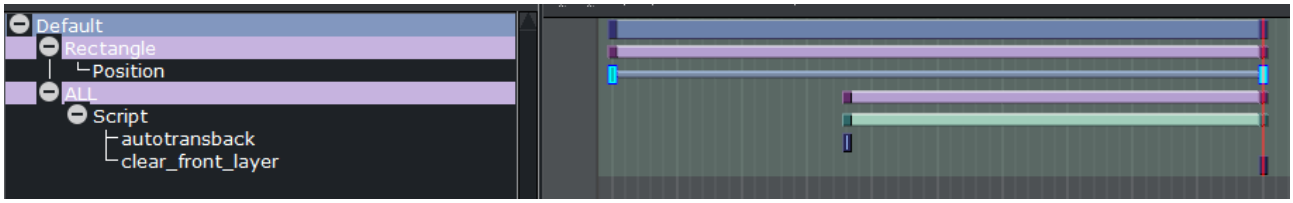
Example Effect

1. The scene is fully transparent at the beginning of its animation.
2. At *Take,* an animation starts to show content, animating to fully cover the scene underneath it (switcher).
3. After a predefined time, the animation reaches the *autotransback* script routine, which triggers the switch in the Switcher Scene.
4. The animation fades out, being fully transparent again.

Script

Two routines, one for triggering the take (auto_trans) and one for its removal when the animation ends:

```
sub OnInitParameters()
    RegisterPushButton("autotransback", "Auto trans back", 1)
    RegisterPushButton("clear_front_layer", "Effect Close", 2)
end sub

sub OnExecAction(buttonId As Integer)
    if buttonid = 1 then autotransback()
    if buttonid = 2 then clear_front_layer()
end sub


sub autotransback()
    dim commandStr as string = "RENDERER*MAIN_LAYER" & "*SCRIPT INVOKE auto_trans"
    println 5, "--------------- TOP SCENE TRANS FROM FL SCENE ----------------"
    System.SendRemoteCommand("localhost", 6100, commandStr, 0)
end sub

sub clear_front_layer()
    dim commandStr as string = "RENDERER*MAIN_LAYER*SCRIPT INVOKE clear_front_layer"
    println 5, "--------------- TOP SCENE OUT ----------------"
    System.SendRemoteCommand("localhost", 6100, commandStr, 0)
end sub
```



Note: Using effects scenes on a *separate/external* Viz Engine.

- Edit the script to send the RemoteCommand for "autotransback" to the Engine Switcher. In this case the IP needs to be manually changed to the correct external one in the effect scene.
- Keep in mind that using an external engine for triggering effect transition switches might lead to timing inaccuracies due to possible network delays.

Custom Content Transitions

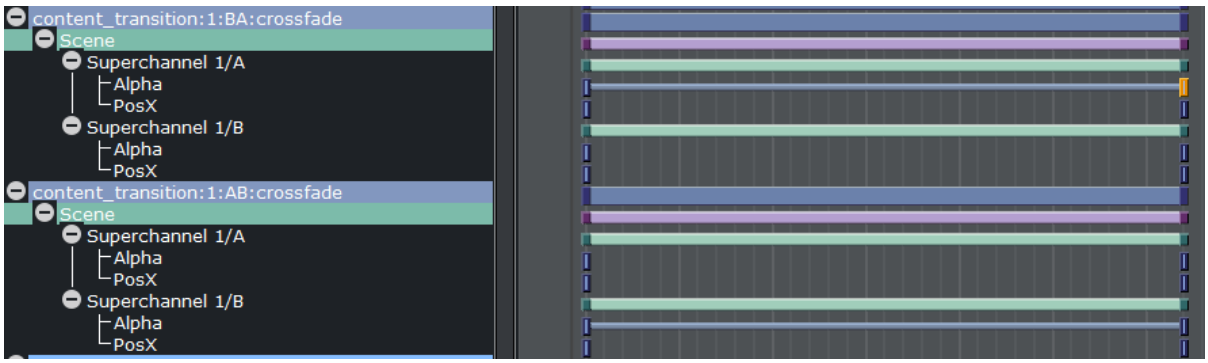
This section describes the actions required in Viz Artist, when working with Engine Switcher-driven Viz Mosart templates.

Note: Content transitions are only used for *full screen* content transitions. Transitions in between Boxes / Splits are always hard cut.

Preset Plugin

The Preset plugin creates a default content transition for a crossfade. Additional transitions for changing content within superchannels are created manually using Viz Artist.

Default Preset plugin crossfades



- A custom content transition is saved to a special director name, called by the Switcher from Viz Mosart. This director can contain the required transition animation.
- Typically, a content transition director contains the keyframes related to the superchannels sub-channels *A* and *B*.
- Content transitions are created in both directions. A separate director is used for a transition between A to B, and B to A.

To define a director

- A director has the following naming format:
  `content_transition:<ID>:<direction>:<transition_name>` where
  - ID: Superchannel ID
  - direction: *AB* or *BA*
  - transition_name: Name the transition that will be called in the UI (for example *wipe*).

Custom Effects

- Custom effects scenes are manually added to the Engine Switcher's configuration file.

> **Note**: Custom effects IDs start at *80*.

To create an Effect

1. In Viz Mosart Mosart AV Automation:
   A/V Setup (c:\channeltemplates\avconfig.xml)

   File Edit **Effects**

	Effect name	Effect Nr	Emem Nr Preview	Emem Nr Program	Emem Nr Cleanup	Duration
▶	TRANSITION	4				10
	HARDCUT	5				10
	EFFECT61	6		61		10
	EFFECT53	7		53		10
	WIPE	80		80		10

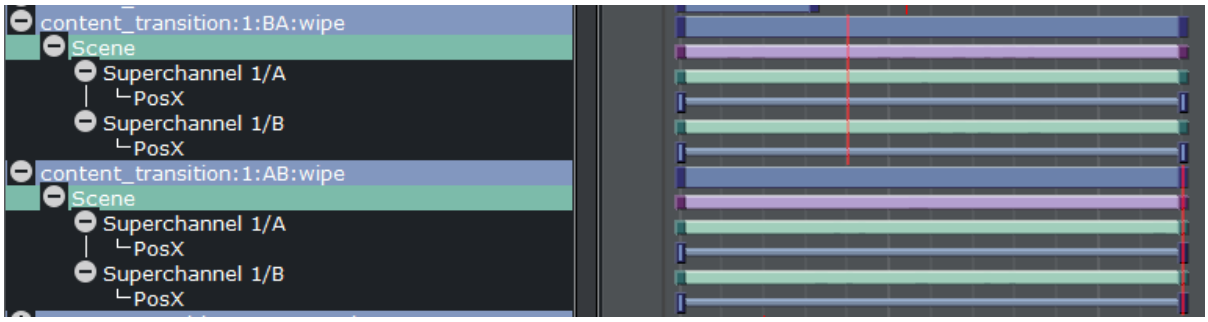
2. In the Viz Engine Engine Switcher Configuration file, C:\ProgramData\vizrt\VizEngine\switcher_config.cfg

```
#custom content transition effects
<content_transitions>80:wipe,81:rotate,82:twist</content_transitions>
```

3. The effect is now available for selection in Viz Mosart:

Example - Effect 80 - Wipe:

A typical wipe is set up by animating the subchannel's X position from left to right, then having the other subchannel follow.



Values:

(1) Direction AB (content_transition:1:AB:wipe):

- Assuming sub-channel 1/A is visible and should fade out to the right
- Keyframes for Pos X are set to
    - Superchannel 1/A:
    - Start : 0.0
    - End : 100.0

- Sub-channel 1/B starts on left to 1/A and animates onto the screen
    - Superchannel 1/B:
    - Start : -100.0
    - End : 0.0

(2) Direction BA (content_transition:1:BA:wipe):

The roles are now switched, this means 1/B starts on the screen, animates to the right and 1/A now appears from the left.

- Keyframes for Pos X are set to
    - Superchannel 1/A:
    - Start : -100.0
    - End : 0.0
- Sub-channel 1/B starts on left to 1/A and animates onto the screen
    - Superchannel 1/B:
    - Start : 0.0
    - End : 100.0

Labels

Label names (newsroom tags) and their content, defined in a Viz Mosart template and filled out in the NCRS, are stored in *shared memory* when the story item containing the box template is cued.

When a new DVE box is cued, the designer dynamically reacts to any label change. These changes can either come from a script included in the Switcher scene background or from another scene located on another networked Viz Engine.

Label Format

The format of the received string comprises label names and label values. Each label pair is separated with a "|" character, giving:

```
<label_name1>:<label_content1>|<label_name2>:<label_content2>|
<label_name2>:<label_content3>|<label_name4>:<label_content4>...
```

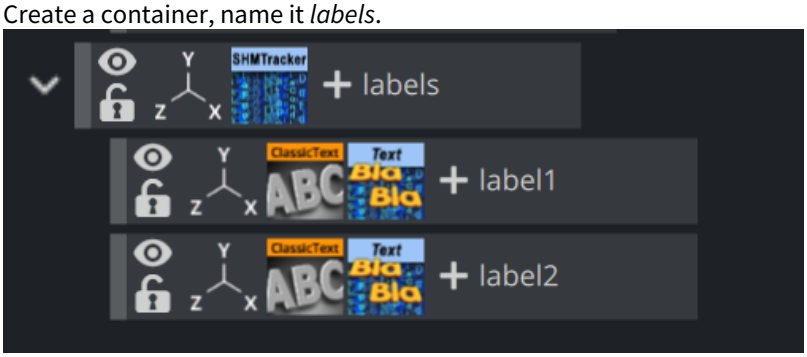
Example: Labels populated from a DataPool plugin

The default map name is *MIXER_LABELS*. This can be changed within the Engine Switcher Configuration File.

In this example you use a Vizrt **Datapool** plugin to access the content of the shared memory and deliver its content to a specified text container.

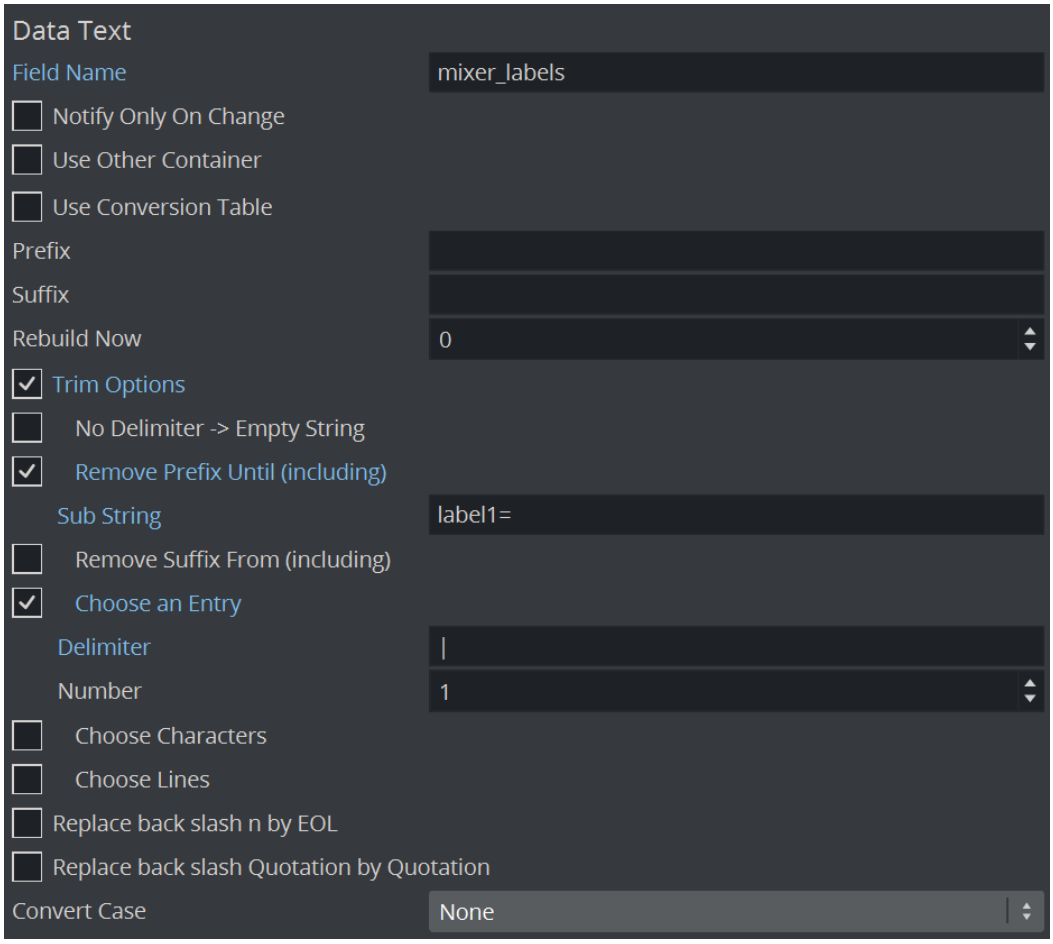
- An *SHMTracker* plugin reads the memory content.
- The content is forwarded to any other **Datapool** plugin within the scene.
- Here, *ataText* receives the memory content from *SHMTracker* and parses the values into a readable format
- The content is pasted into the geometries **Text** plugin.

1. Create a container, name it *labels*.



2. Add the *SHMTracker* plugin to it.
3. Set a unique ID for **Output Field Name**, in this example *mixer_labels*.

4. To access the Switcher scenes used SHM, set **Shared Memory Type** to *Distributed*.
5. Assign **Share Memory Key Name**. Default is set to *MIXER_LABELS*.
6. Create a child container and name it *label1*.
7. Add the *Data Text* plugin.
8. Set **Field Name** to the same *ID as* used in the parent plugin. In this example, *mixer_labels*.
9. In **Rebuild Now**
    a. Check **Trim Options**.
    b. Check **Remove Prefix Until** and enter **Sub String** *label1=*
    This crops the *label* from the incoming string.
    c. Check **Choose an Entry**.
        i. Set **Delimiter** to |
        This crops the *string*, revealing the first label entry.



10. Repeat these steps for all required labels, modifying the sub-strings accordingly to *label2=* etc.

Example of a container script

The default map name is *MIXER_LABELS*. This can be changed within the Engine Switcher Configuration File.

This simple script example shows how to read in the content of a shared memory map, parsing the content for *label name* and *content* and pasting it onto an existing text container.

The given example reacts on a mapping change that typically is triggered from cueing the next *Box* item.

```
'-------------------------------------------------------------------------------------
-----------------------------------
sub OnInit()
    VizCommunication.map.Registerchangedcallback("MIXER_LABELS")
end sub

sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
    dim labels as String = VizCommunication.map["MIXER_LABELS"]

'parsing and reacting routines
    Parse_Labels(labels)
    Set_Labels()
```

```
end sub

sub Parse_Labels(input as String)
    Dim params as Array[String]
    Dim content as Array[String]

    input.Split("|",params)

    For each param in params
        param.Split(":",content)
        labels[content[0]] = content[1]
    Next
end sub

'setting labels (container name equals label name)
sub Set_Labels()
    dim label_keys as Array[String]
    labels.GetKeys(label_keys)

    for each key in label_keys
        This.FindSubContainer(key).Geometry.Text = labels[key]
    next
end sub

'-------------------------------------------------------------------------------------
-----------------------------------
```

Note: For more advanced use cases, please contact Vizrt Support.

Note: The actual toggle event of the next story item can be accessed by registering on the the *<toggle_status>* map, as defined in the Engine Switcher Configuration File.
This allows the designer to trigger animations related to an upcoming change of the layout.

NDI Clip Preview

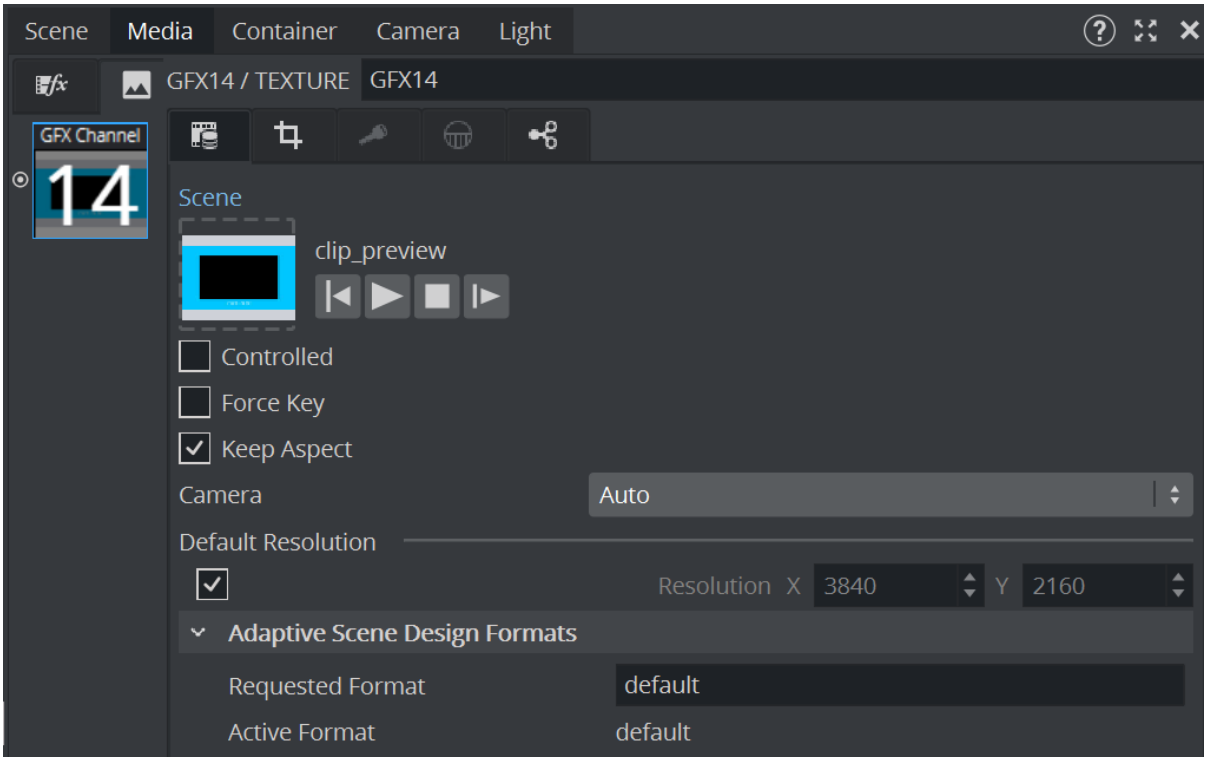
If a simple preview of the upcoming clip layout/switch is needed and a separate Viz Engine cannot be used, you can instead add a small nested scene to the Switcher scene. This can stream the next cued clip via NDI (Network Device Interface, a protocol).

Note: The *Media Preview* feature used in Viz Opus is no longer supported.

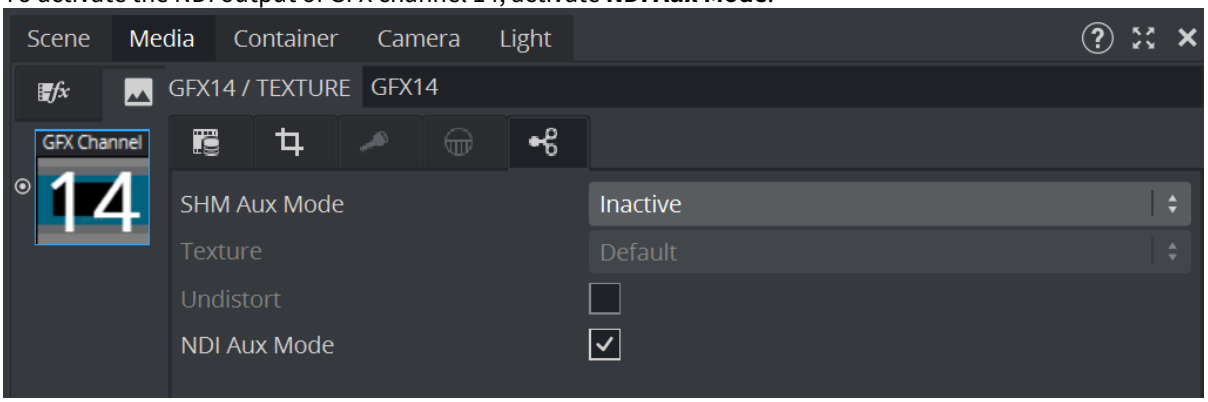
To stream a preview with NDI

Engine Switcher is configurable to send clip cueing commands to a separate scene if needed.

1. Add the scene to a GFX channel within the Engine Switcher scene.
   By default, GFX channel 14.
2. Add this channel manually to any existing Switcher scene.



3. To activate the NDI output of GFX channel 14, activate **NDI Aux Mode**.



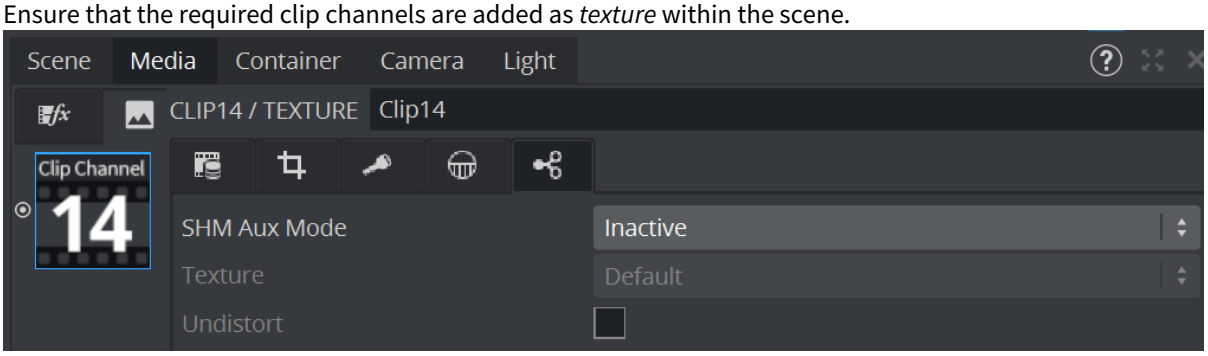
GFX channel 14 will now be received as a separate NDI stream on the network.

Previewing a scene

The preview scene setup is simple.

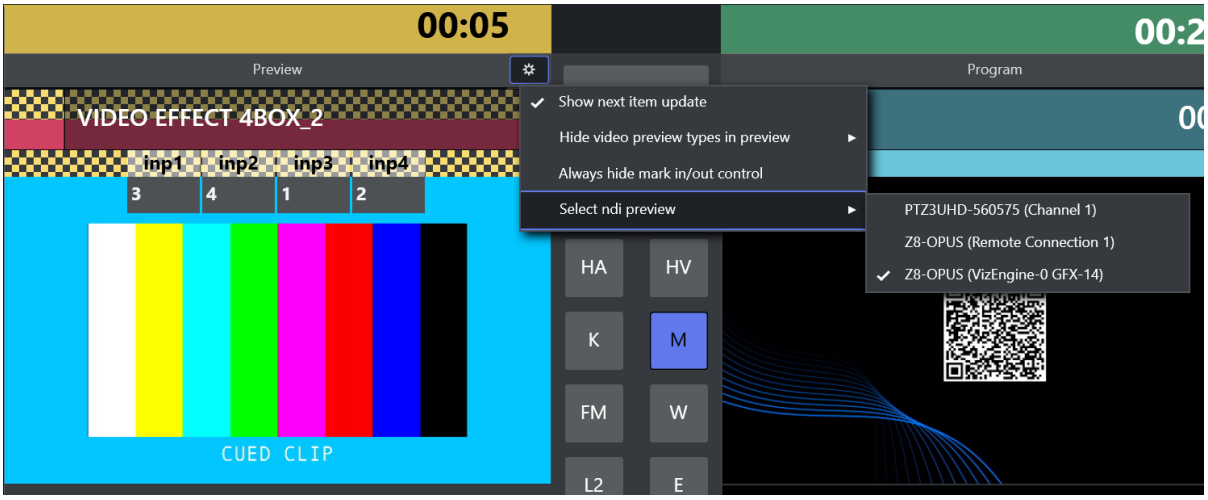
4. Clip channel 14 and 15 are used to load the next cued *full screen* or *box* clip.
   The layout of the scene itself does not matter and requires no further logic.
5. The two clip channels are added as *texture* to the scene and can be placed as required.

6. Ensure that the required clip channels are added as *texture* within the scene.



Setting up Viz Mosart

7. In Viz Mosart, select the source in the Preview window.
8. As soon as a clip is cued as next item, the GFX channel scene loads the paused clip. This updates the NDI stream of the Mosart UI.



- **SUMMARY OF CONFIGURATION POINTS**

  Viz Engine Configuration

  - C:\ProgramData\vizrt\VizEngine\VizEngine-0.cfg
    (**Note**: Here -0 refers to which GPU will be used).

> **Tip**: A copy of this config is included in this Administration Guide under Device Configuration Files > Engine Switcher Configuration File.

Switcher scene configuration

> The separate config file "switcher_config_<versionNumber>.cfg" **must** be renamed to drop the versionNumber, so that it just reads "switcher_config.cfg".
> For example, if the default name is *switcher_config_0.cfg*, this file must be renamed to *switcher_config.cfg*.

- C:\ProgramData\vizrt\VizEngine\switcher_config.cfg

Viz Media Sequence Configuration

- localhost:8580/app/vdomconfig/vdomconfig.html

Viz Mosart

- C:/channeltemplates/VizEngineSwitcherConfig.xml
- C:/channeltemplates/ClipServerVizEngine.xml

> **Note**: VizEngineSwitcherConfig.xml and ClipServerVizEngine.xml are originally located in the Viz Mosart installation folder C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles.

See Also

- AV Automation Devices Vision Mixer

Clean and Dirty-Feed Setup

By default, Engine Switcher is set up to show only a combined dirty feed output. A clean feed configuration must be set up manually.

> Clean and dirty feed
>
> - A video signal that does not have any graphics or effects embedded, is called a *clean feed*.
> - A video signal that has graphics or effects enabled is called a *dirty feed*.

- Working with Clean and Dirty Feed
  - Engine Switcher Terminology
  - Hardware Configuration
  - Software Configuration
  - Configuration Example
  - Viz Mosart Overlay Graphics Configuration
  - Additional Software Configuration
- Running **two** Viz Engine instances on the Engine Switcher server enables production of both dirty and clean feed.
- This setup will use one program output on the clean feed and one video input on the dirty feed.
- The clean feed signal is routed back into Engine Switcher through the Matrox video card, using the zero-frame delay at one of the video inputs.
  Zero-frame delay at the input prevents any additional delay to the final program signal.
- All overlay graphics and transition effects are then added on the clean feed signal, producing the dirty feed.
- The dirty feed is then sent out on two program-out channels.

> **Note**: Clean and Dirty feed are rendered by 2 instances running on the *same* GPU. As resources are shared, creating a heavy load on one instance might affect the performance of the second one.

Working with Clean and Dirty Feed

Engine Switcher Terminology

Clean Feed

The clean feed provides:

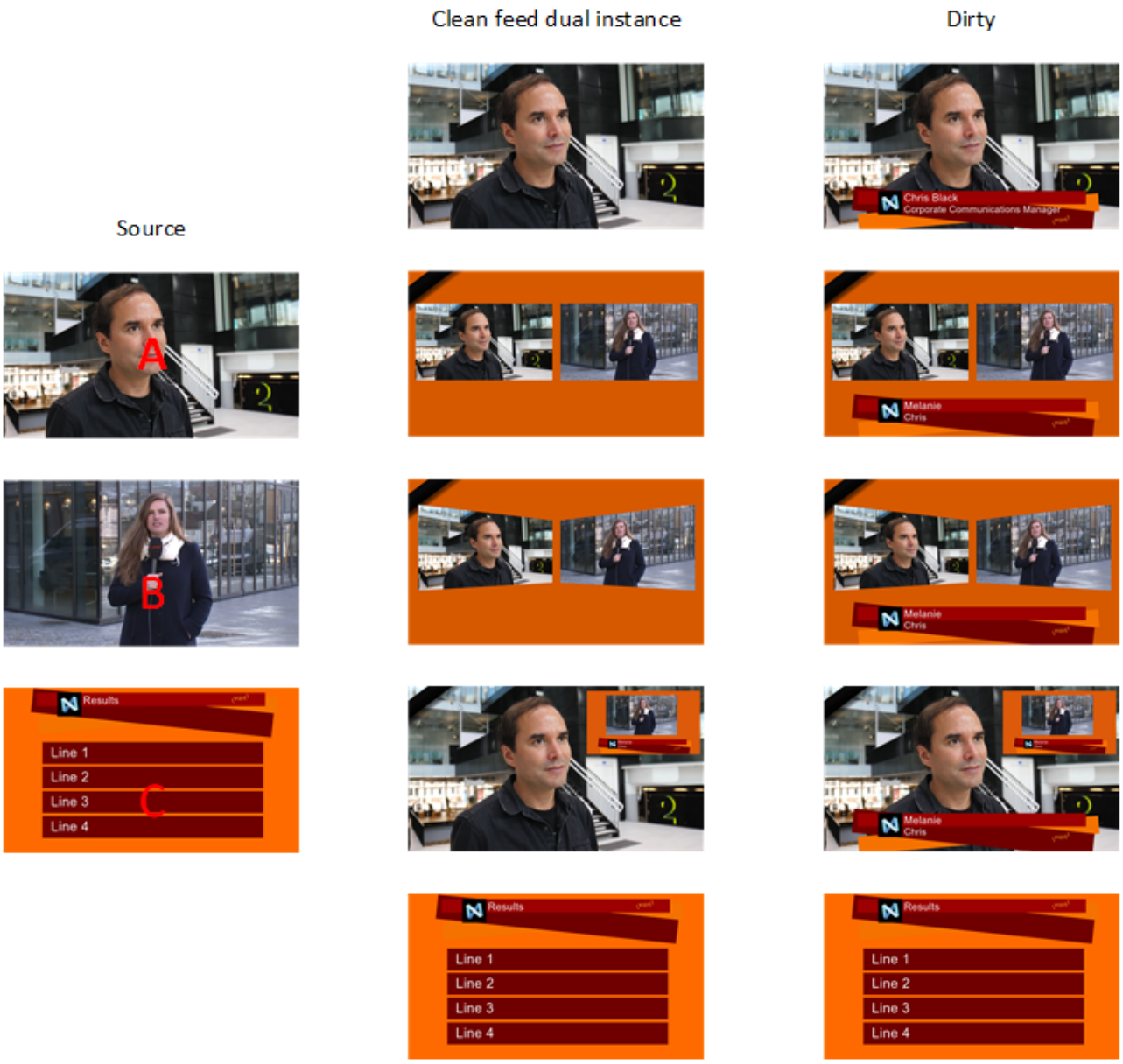
- Full screen video live sources
- Full screen clip sources
- Full screen GFX elements
- Butterfly effects (PiP effects)

Dirty Feed

The dirty feed adds overlay graphics on top of the clean feed output.

	Clean Feed Engine	Dirty Feed Engine
Front Layer	Transition Effects	Overlay scenes
Main Layer	Full screen GFX	Overlay scenes
Back Layer	Not used	

Example: Output of clean and dirty feed.

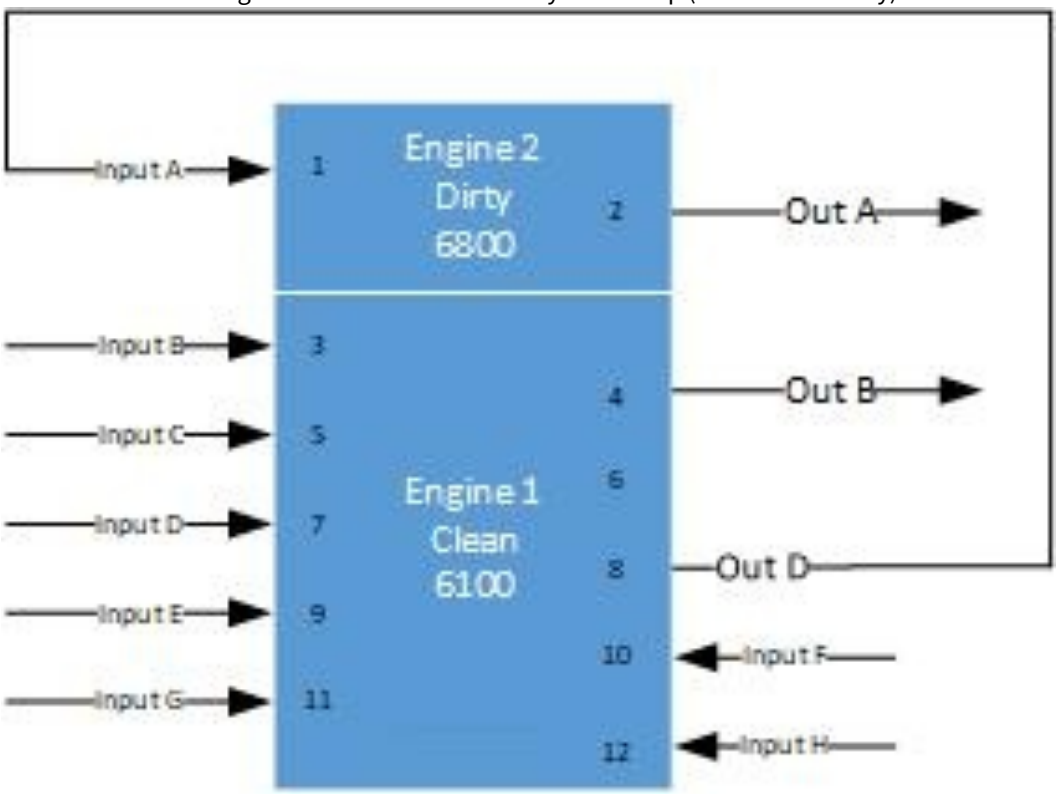


Hardware Configuration

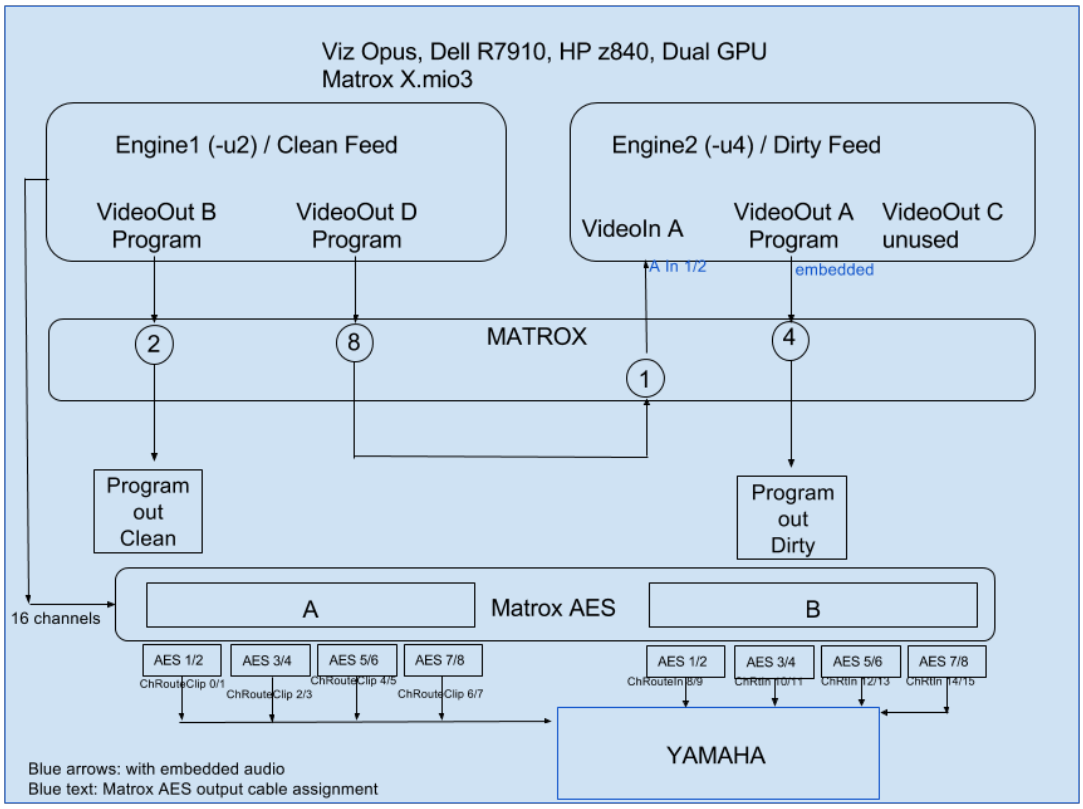
The current clean feed solution needs a hardware wiring using a short SDI cable from the SDI OUT D (pin 8 on the X.mio3) to SDI IN A (Pin 1). Engine Switcher is using the "Zero Frame Delay" option within Viz Engine 5.1. This provides a direct signal copy from Input A to Output A without introducing an additional delay. Therefore there is no need to load a keyer scene to the dirty feed engine.

To configure the Matrox board

- Matrox Pin Out for Engine Switcher clean and dirty feed setup (Zero Frame delay):



- Routing of video signal on the Matrox board.



Software Configuration

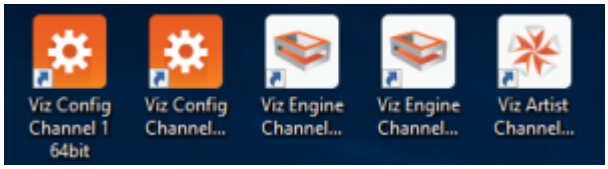
This sets up Viz Engine in dual channel mode.

To configure the Viz Engine in dual channel mode

Install Viz Artist and select **Dual Channel** mode.

> **Tip**: To create the necessary shortcuts, refer to the *Dual Channel Mode* section in the latest version of the Viz Engine Administrator Guide (here as an example, version 5.1).

Dual Channel installs and adds these icons to the desktop:



1. Replace the default configuration files with those provided on the Vizrt FTP.
   Files are in the folder /Product/VizMosart/Engine Switcher Scenes/

2. Rename them according to the computers name:
    a. Engine 1 - clean feed -> <machinename>-0.cfg
    b. Engine 2 - dirty feed -> <machinename>-2.cfg
3. Modify the dual channel shortcuts (different scenarios related to GPU count mentioned above).
   For Media Preview to work in Engine Switcher the clean feed instance needs to run in *console mode* (no Viz Artist).
    a. Configure the desktop shortcuts properly by adding parameters:
        i. -x (no UI, *mandatory*)
        ii. -u1 or -u3 depending on the graphics board used.
            1. -u1 points to the first engine on the first GPU (used for clean feed)
                a. this uses -> <machinename>-0.cfg
            2. -u3 points to the second engine on the first GPU (used for dirty feed)
                a. this uses -> <machinename>-2.cfg
        iii. -n for engine mode
        iv. -c config mode
            - u1 or -u3 for config shortcut of engine 1 or 2.
4. Port Configuration.
   Make sure that the ports are set correctly. This should be set when using the
    a. Clean feed runs on port 6100
    b. Dirty feed runs on port 6800.

Configuration Example

To configure single GPU dual engine setup

Since all engines do use the same GPU, the "-u" parameter in the shortcut is only relevant to point the engine to the right config file. A clean- and dirty channel setup typically will use:

1. **Engine 1**: Clean feed
    a. <machinename>-0.cfg
    b. Desktop shortcut uses "-u1"
2. **Engine 2** : Dirty feed
    a. <machinename>-2.cfg
    b. Desktop shortcut uses "-u3"

To configure dual GPU dual engine setup

Typically the Engine Switcher's dual engine setup comes with two GPUs. The first card is used for connecting the monitor, the second for actual rendering purposes.

Config files and desktop shortcuts will have to be named as follows:

1. **Engine 1**: Clean feed
    a. <machinename>-1.cfg
    b. Desktop shortcut uses "-u2"
2. **Engine 2**: Dirty feed
    a. - <machinename>-3.cfg
    b. - Desktop shortcut uses "-u4"

Note:

- The digit behind the "u" depends on how the graphics board is installed in the machine.
  If there are 2 GFX boards, the weaker one is used for the monitor, not for rendering. This weaker board is typically not addressed.
  So the second video boards gets -u2 (clean) and -u4 (dirty).
- Since Viz Artist always runs on the *first* GPU, a third config file named <machinename>-0.cfg must be employed, in case changes in the scene have to be performed on this particular scene.
- As Viz Artist will run on the weaker GPU, performance of the scene while being loading on this Engine will be limited.

Output Format

Please note that the video format must be manually modified for the affected config files.

- The default is set to **1080i50.**

To set the output format

Changing the format is most easily done by loading the appropriate configuration profiles.

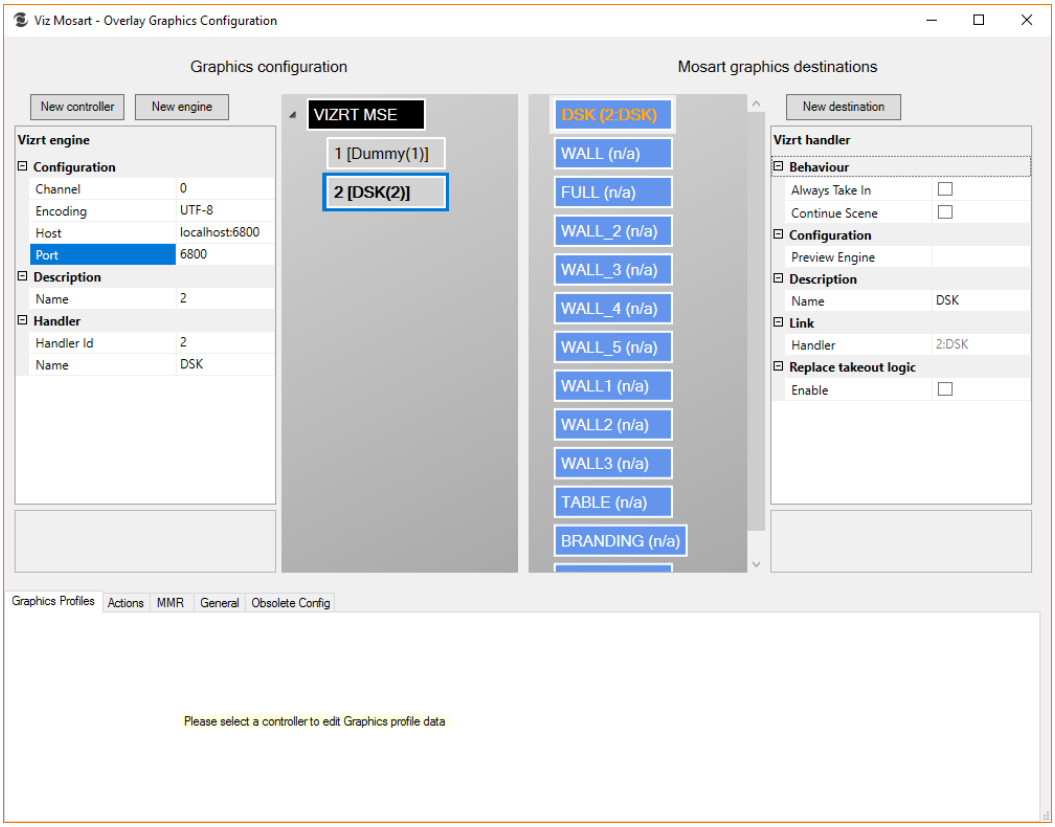
1. Start the Configuration Tool for the clean feed Viz Engine.
2. Click **Load** and choose **Installed Profile**.
3. Navigate to C:\Program Files\vizrt\VizEngine\Configuration Profiles and choose the correct option
   - video-resolution-1080i50-audio-xmio3.cfg
   - video-resolution-1080i60-audio-xmio3.cfg
   - video-resolution-720p50-audio-xmio3.cfg
4. Save the configuration and close the Tool.
5. Start the Configuration Tool for the dirty feed engine and repeat steps 2-4.

Viz Mosart Overlay Graphics Configuration

To set up Overlay Graphics configuration

The Overlay Graphics applications configuration needs modifying, to send Overlay scenes to the dirty feed on Port 6800 (instead of the default port 6100).

1. From the Viz Mosart server machine, open the Mosart Graphics Interface application (MMOverLayGraphicsInterface.exe).
2. From the menu bar, select **Settings > Properties**.
3. Fill-in as illustrated:



4. Restart the applications to save changes.

Additional Software Configuration

Configure the Viz Engine in dual channel mode, as described in the *Viz Engine Guide*, section *Software Configuration > Dual Channel Mode*.
Whilst this can initially be installed as a single engine, you will still need to configure the shortcuts and adjust the Viz Config.

Execute all three procedures:

1. Set dual channel mode (Viz Engine)
2. Configure clean feed
3. Configure dirty feed

To set dual channel mode (Viz Engine)

1. For default configuration of a dual channel, load the dual channel profiles for Channel 1 (Clean) and 2 (dirty). This will set the correct Ports.
   C:\Program Files\vizrt\VizEngine\Configuration Profiles
   - dualchannel -0.cfg
   - dualchannel -1.cfg
2. Load the correct audio profile on both channels via the configuration tool
   C:\Program Files\vizrt\VizEngine\Configuration Profiles
   - video-resolution-1080i50-audio-xmio3.cfg

To configure clean feed

The clean feed configuration should be based on the file provided in the bundle installer.

1. Configure **Program** and **Contains Alpha** to *Inactive*.
   See Video out documentation to configure the program output of the engine to output program (fill) on VideoOutB and VideoOutD assigned by setting the Map to Viz channel t for reference.
2. Configure the video inputs.
   a. Start by disabling video 1 by setting VideoInA to Unused.
   b. Set all other to the corresponding Video number
      (VideoInB to Video2, VideoInC to Video3 etc).
3. Changes done with the configuration tool are stored in `<machinename>-1-0.cfg`.
4. **Video** (Matrox board configuration)
   a. Mapping Live Outputs:
      i. Set Out B and Out D to *Program*
      ii. Set Out A and Out C to U *unused*
      iii. Set In A to *unused*
   b. Mapping Live Inputs:
      i. In A set to *unused*
      ii. In B set to *Live 1*
      iii. In C set to *Live 2*
      ...
      iv. In H set to *Live 7*
   c. Activate Live 1 to Live 7 in Input config.
5. **Audio**
   a. Audio Mode is set to AES
   b. AES audio routing for clips needs to be done directly in the config file.
      Default settings for 2 separate audio channels:

ChannelRouteClip0_0	0
ChannelRouteClip1_0	2
ChannelRouteClip0_1	1
ChannelRouteClip1_1	3

To configure dirty feed

See the *Viz Engine Guide*, section *Software Configuring Viz > Video Output* for reference.

1. Changes should be done uses the Configuration Tool.
2. Configure the program output of the engine to output program (fill) on VideoOutA assigned:
3. Set **Map to Viz channel** to *Program*
4. Set **Contains Alpha** to *active*.
5. Configure the video inputs by setting all of them to *Unused*.
6. If not already set by the preloaded dual channel profiles, set communication **General communication port** on this engine to *6800*.

7. Verify that the changes are stored in `<machinename>-3-0.cfg`.
8. **Video (Matrox board configuration)**
    a. Mapping Live Outputs
        i. Set Out A to Program
        ii. Set Out B, Out C and Out D to *Unused*
    b. Mapping Live Inputs
        i. Set all Inputs to *Unused*
9. **Audio**
   Audio on the dirty feed is *not* supported by default.
   Routing of the AES channels is performed manually in the Viz Engine's configuration file.
10. **Additional settings**
    - Turn on *Zero Frame Delay* option.

12.1.4 Avid iNews Web Service for Status Feedback

This section shows how to set up and use Viz Mosart with Avid iNEWS Web Services in order to get stories status updates in the NRCS (Newsroom Computer System). This section covers the following topics:

- Introduction
- Supported Versions
- iNEWS Web Service Installation and Configuration
  - Avid iNEWS Web Service
  - Determining the Version of iNEWS Web Service
  - Configuration
  - API Session Limits
- Viz Mosart Installation and Configuration
  - Web Services Connection
  - Other Manus Administrator Settings affecting NRCS Status Feedback
  - Configurations in AvidInewsWebServiceConfig.xml
- Operation
  - Status for Stories Containing more than One Item
  - Status of an Off-air Story Containing Multiple Clips
  - Clip Duration Feedback to iNEWS Server
- Troubleshooting and Known Issues
- Useful Links

Introduction

Avid iNEWS Web Service can be configured with Viz Mosart connected to iNEWS through FTP. The main purpose is sending status for stories that contain clips (Online/Offline for clip availability, Cued, Play or Stop and duration, as provided by the video server), but the system can also be configured to send status for all types of events (Cameras, DVEs etc.).

> **Tip**: Another useful Viz Mosart utility, the *iNEWS Timer* provides accurate timing of the production in iNEWS by sending a shortcut to the iNEWS client each time an item is stepped through in the Viz Mosart story. See the description under Viz Mosart clients.

Supported Versions

iNEWS WebService	Tomcat	JDK	iNEWS server
1.6.0.5	6.0.37	1.8	up to 2022.12
1.6.0.5	up to 9.0.79	20	up to 2022.12
1.6.3	up to 9.0.79	20	up to 2022.12

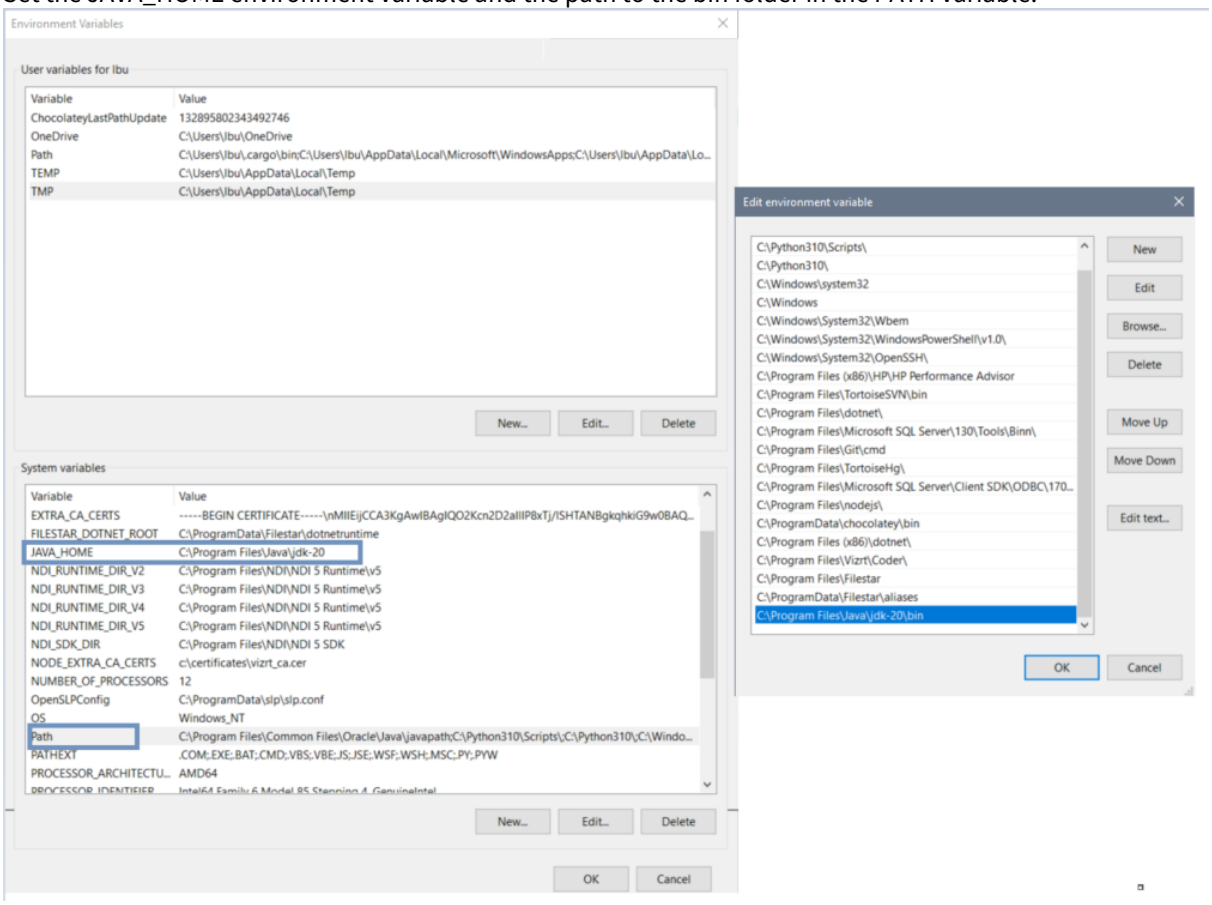
iNEWS Web Service Installation and Configuration

Avid iNEWS Web Service

To install the Avid iNEWS Web Service

> **Info**: Refer to the documentation that comes with Avid iNEWS Web Service installation package.

1. From oracle.com (Windows offline (64-bit)), download and install the latest JRE or JDK.
2. Run the executable JRE installation file (for example, *jdk-20_windows-x64_bin.exe*).
3. Set the JAVA_HOME environment variable and the path to the bin folder in the PATH variable:



4. From tomcat.apache.org, download and install the latest stable 32-bit/64-bit Windows service installer of Tomcat (for example, *apache-tomcat-9.0.79.exe*).
5. Run the executable Tomcat installation file:
   a. Select “Normal” installation type.
   b. Create a Tomcat Administrator login user name and password (Do not name the user *tomcat*).
   c. Select the JRE installation path if it is not already selected (for example, *C:\Program Files\Java\jdk-20*).
6. Allow C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\Tomcat9.exe through the Windows Firewall.
7. After the Tomcat installation is finished and has started, navigate to the Tomcat management page at http://<server>:8080/manager/html.

8. Log in from the Tomcat Management page, with username/password you defined at step **5** above.
9. The default connector port (8080) can be modified in Tomcat's .\conf\server.xml file.
   If port 8080 is used by another applications, edit the Tomcat Installation/config/server.xml file:

```
<Connector port="8080" protocol="HTTP/1.1"   connectionTimeout="20000"

redirectPort="8443" />
```

10. After any port changes, from Services.msc or from Start > All Programs > Apache Tomcat > Monitor Tomcat, restart the Tomcat service.
11. An icon is created in the Windows Taskbar.
    Right-click the icon for a list of actions to Start and Stop Tomcat service.
12. Deploy iNews web service.war
13. After deployment, verify that the iNEWS Web Service is available at
    http://<server>:<port>/inewswebservice/services.

Post installation actions

If you have problems accessing the Web Services page (http://<server>:<port>/inewswebservice/services), try deleting the Tomcat cache:

1. Stop the Tomcat service.
2. Delete folder Tomcat installation\webapps\inewswebservice (only the folder, not the war file).
   You may replace the war file, for example, with a new one when upgrading. (This solution is same as deploying from Tomcat manager).
3. Delete folder Tomcat installation\work\Catalina\localhost\inewswebservice
4. Start the Tomcat service and reattempt accessing the Web Services page.

Timeout recommendations

- We recommend reducing the session timeout in Apache Tomcat to *1 minute*, which frees-up the API session 1 minute after Manus Administrator is closed.
- If Manus Admin is closed and opened more than maximum API sessions allowed in 1 minute, it will reconnect automatically to the Web Service after 1 minute, when at least one session has been released. To change the session timeout follow this procedure:

To change the Apache Tomcat session timeout interval

Info: Tomcat configuration files are located at: C:\Program Files\Apache Software Foundation\Tomcat 8.5\conf.

1. Stop the Tomcat service.
2. Change Tomcat session timeout interval in Tomcat installation/conf/web.xml to 1 minute:

```
<session-config>
          <session-timeout>1</session-timeout>
</session-config>
```

3. Start the Tomcat service.

Determining the Version of iNEWS Web Service

The version number should match the version of Avid iNEWS web service you are running. Follow this procedure to get the version number:

To determine iNEWS Web Service version

1. Navigate to the Apache Tomcat installation folder at /webapps/inewswebservice/META-INF.
2. Open the file Manifest.mf.
3. Check the Implementation-version.
4. Recommended version is *1.6.0.5* or *1.6.3*.
   It should look like this:

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.8.4
Created-By: 1.7.0_25-b17 (Oracle Corporation)
Implementation-Build: MSN_INEWS_WEB_SERVICE_1.6.3.1
Implementation-Title: Avid iNEWS Web Service
Implementation-Version: 1.6.3.1
Implementation-Vendor: Avid Technology, Inc.
Copyright: Copyright (c) 2009-2022 Avid Technology, Inc.
```

Configuration

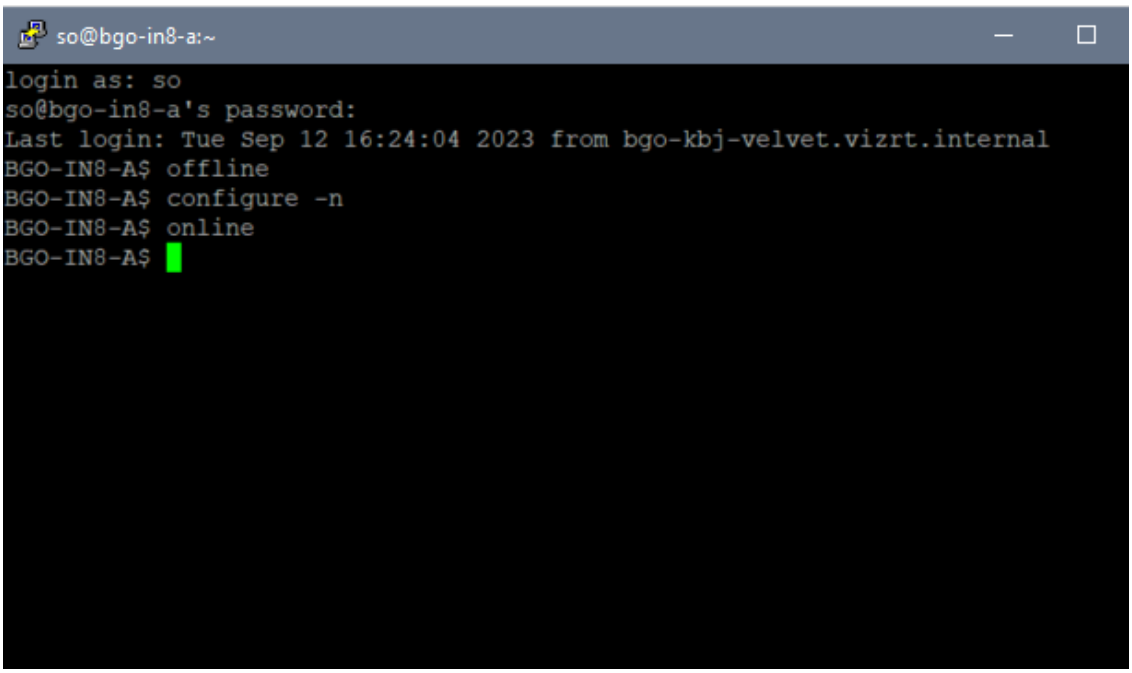
Info: iNEWS can only display information at *story* level and therefore rules have been defined, when a story contains more than one clip.
However, to optimize this feature, it is recommended to have stories containing only *one* clip.

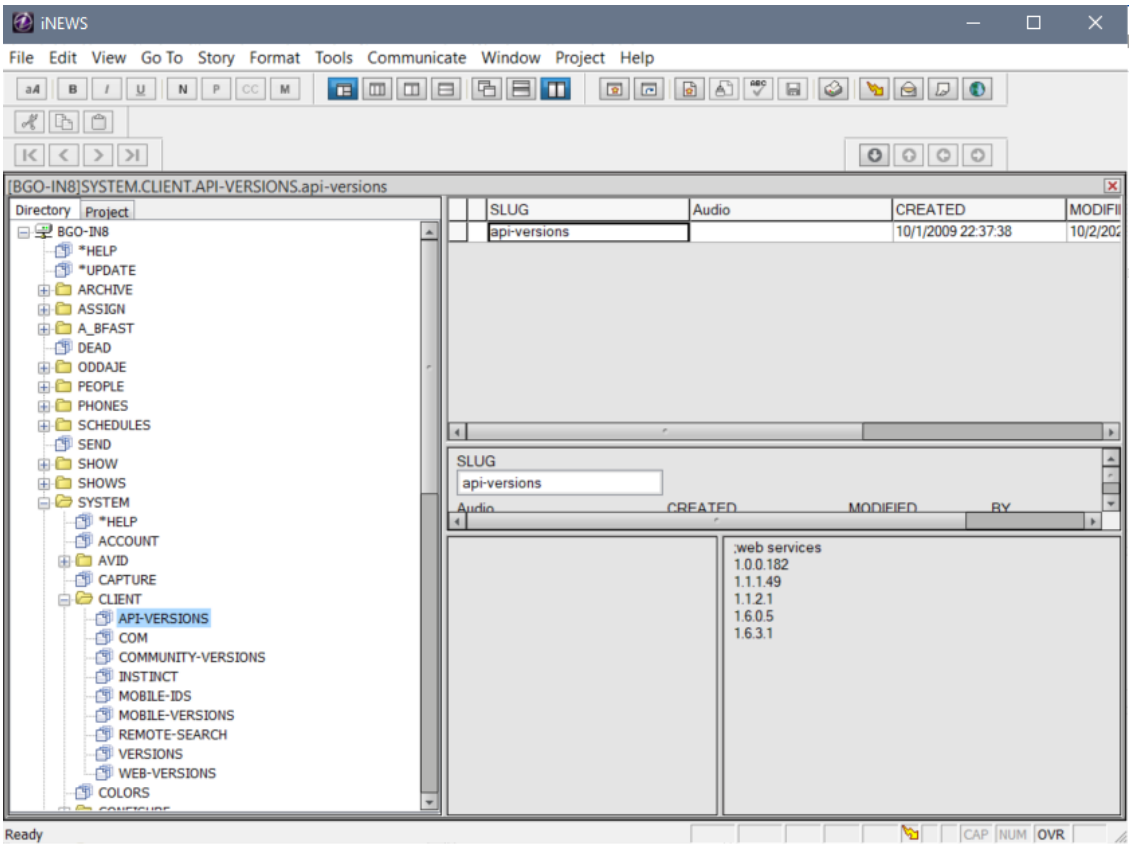
1. Open iNEWS client and add the Avid iNEWS Web Service version in first story of rundown SYSTEM.CLIENT.API-VERSIONS.
   The version number should match the version of Avid iNEWS web service you are running (see section Confirming the Version of iNEWS Web Service).
   For example:

```
;web services

1.0.0.182

1.1.1.49

1.1.2.1

1.6.0.5

; run offline; configure -n; online after doing changes in iNews server console
```

2. On the iNews server console, execute the following commands one by one:
   a. **offline**
   b. **configure –n**

c. **online**

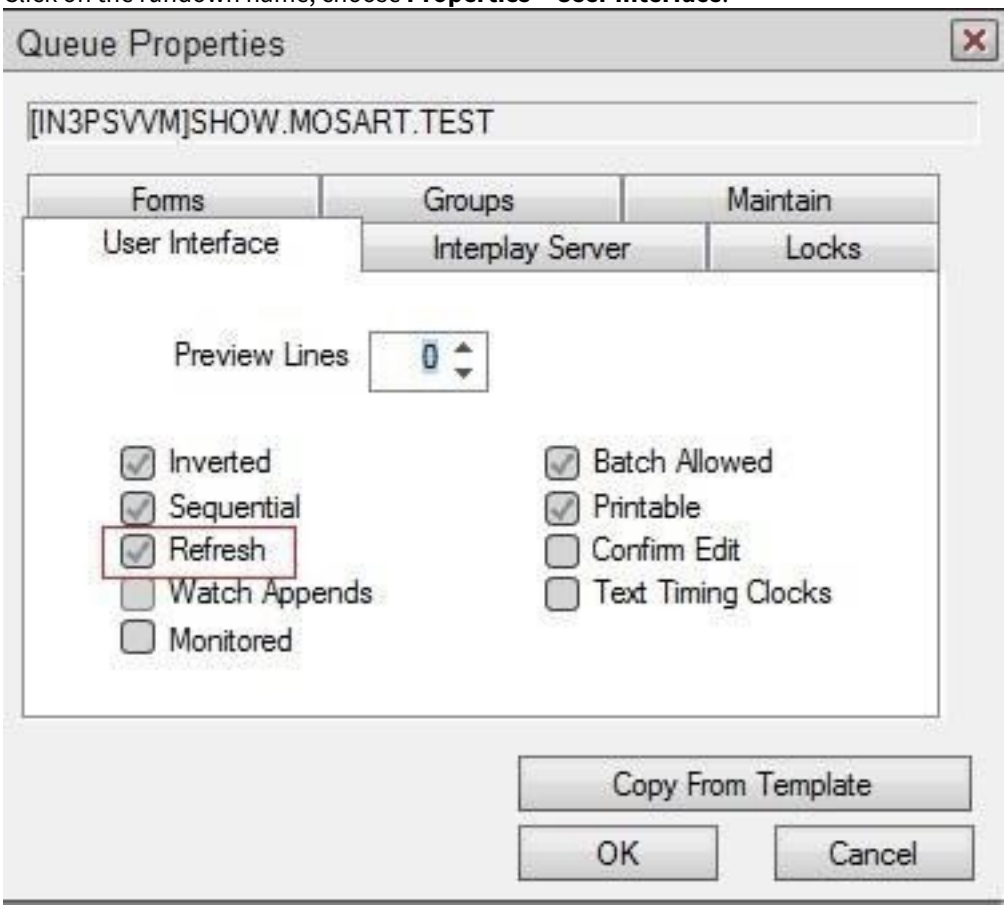




3. For iNews Web Service version >= 1.6.3 (i.e. Mosart Server version >= 5.4.0), *disable* the version check on the iNews server

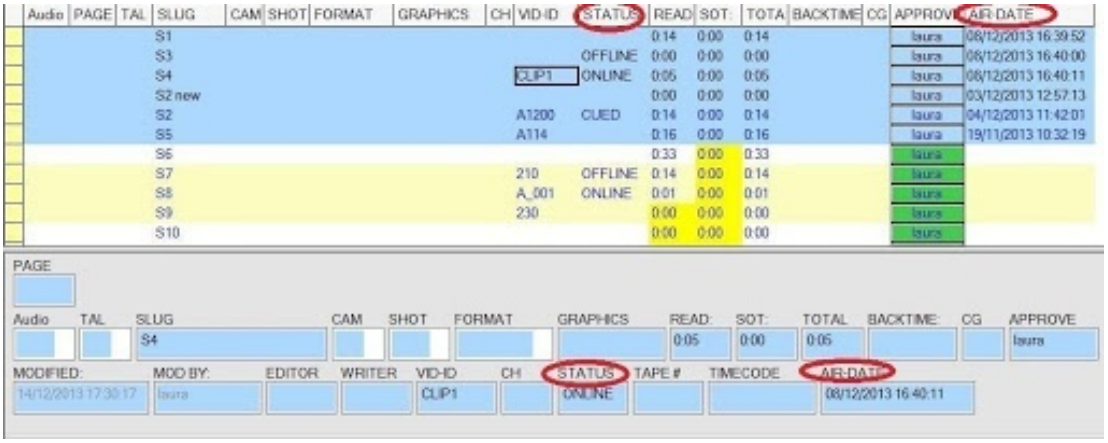
Execute the following command at the console:
`workdebug nocheck noaddress`.

4. Enable Refresh for the rundown used for receiving status updates.
5. Click on the rundown name, choose **Properties > User Interface**.

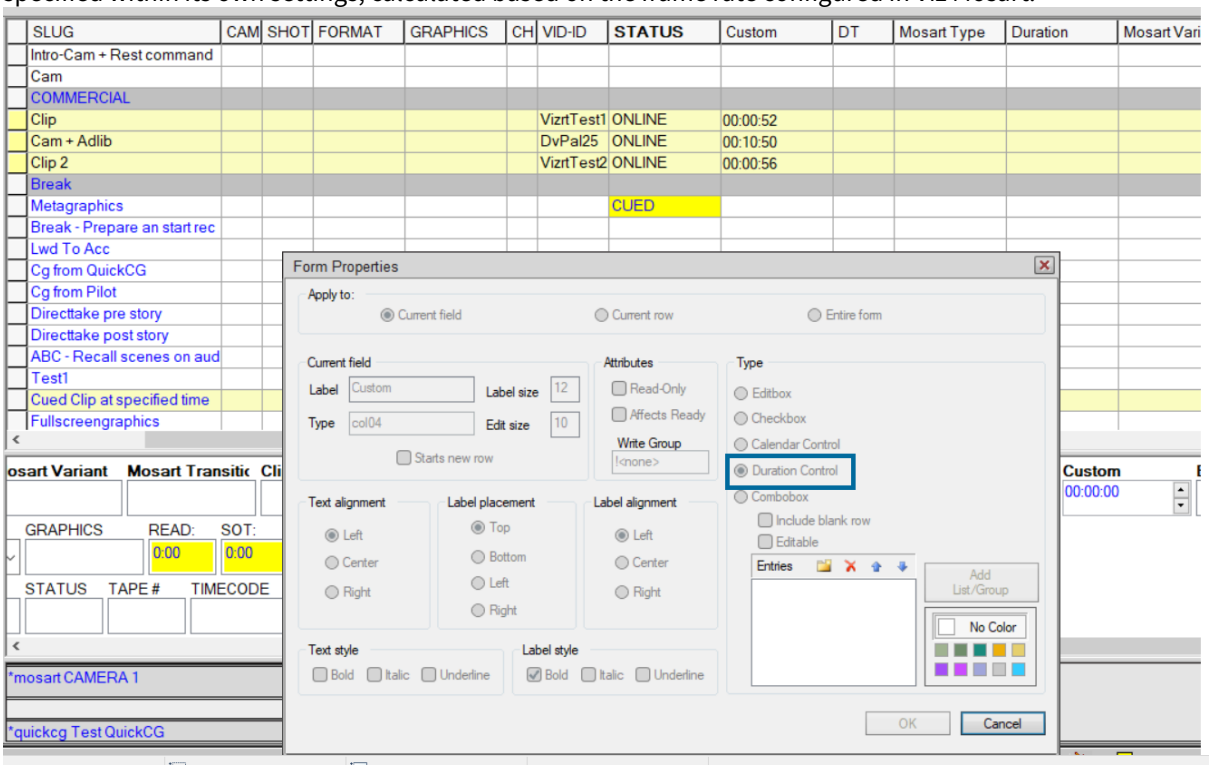


Info: From the iNEWS Setup and Configuration Guide: *Use the Refresh trait only on important queues, like rundown queues that are often modified by multiple users simultaneously.*
To automatically refresh a queue, your system must spend a lot of time monitoring workstations where users are viewing that queue.
Assigning the refresh trait to too many queues that are often accessed at the same time greatly increases the amount of work your system has to do and may severely degrade its overall performance.

6. Configure the columns to receive status and air-date updates from Viz Mosart.
   By default, Viz Mosart uses the field types “event-status” (STATUS) and “air-date“ (AIR-DATE) for this purpose.



7. To enable the feature that allows Viz Mosart to send clip durations, configure a designated column in iNEWS to receive this information.
8. If the column type is set to 'Duration,' then iNEWS will be responsible for controlling the timecode format displayed in that column.
9. Alternatively, if the column type is 'Editbox,' Viz Mosart will send the clip duration in the timecode format specified within its own settings, calculated based on the frame rate configured in Viz Mosart.



API Session Limits

Maximum API-session connections to iNEWS Server

The default license for Avid iNEWS Web Service covers *five* concurrent API sessions.
These are not the same as the iNEWS workstation licenses.

- Type "/exc/list type=api c" at the iNEWS server to see the number configured.
  Additional concurrent API-session licences are available from Avid.
- Viz Mosart will use a maximum of *two* API-sessions (in a redundancy setup).
- If you experience issues with no available API-session, please see section **Troubleshooting and Known Issues**.

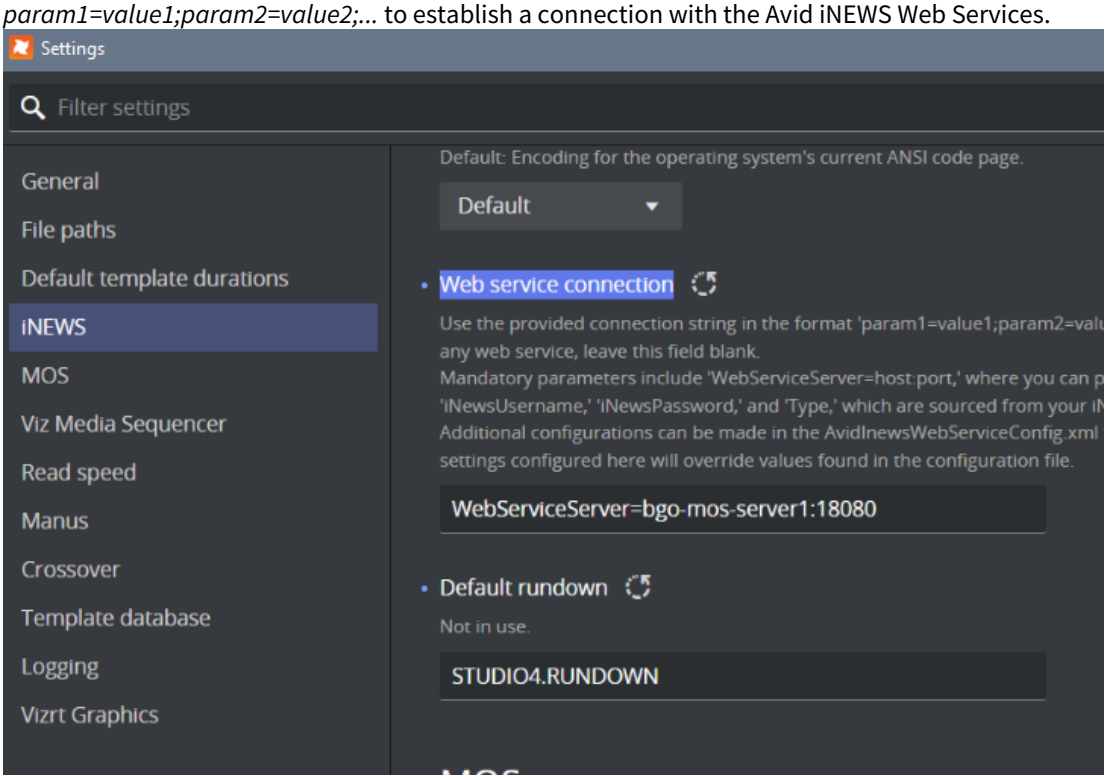
Viz Mosart Installation and Configuration

Web Services Connection

To configure web services

You perform all configurations in one place, from the common Viz Mosart **Settings** menu.

1. From the Manus Admin Console, type *settings*.
2. From the **Settings** menu, in the left-hand panel, select **iNEWS**.
3. In the Web service connection field enter a connection string in the following format *param1=value1;param2=value2;...* to establish a connection with the Avid iNEWS Web Services.



   Leave the field blank if you do not wish to connect to any web service.

Required parameters are:

- **WebServiceServer =** host:port, where the host is the machine where the iNews Web Service was installed (see the iNews Web Service Installation section above). The host can only be provided if the default port (8080) is used.

Optional parameters are:

- **iNewsServer** Only one iNEWS server can be used by the Web Service. Specify which iNEWS server the Web Service should connect through. This field is optional if only one iNEWS server is set in the iNEWS configuration.
- **iNewsUsername** Optional as it can be taken from iNEWS Configuration.
- **iNewsPassword** Optional as it can be taken from iNEWS Configuration.
- **Type** Valid values are 'iNEWS' and 'MosiNEWS

Additional configurations can be made in the AvidInewsWebServiceConfig.xml file, located in the Mosart installation folder under Mosart Server/ConfigurationFiles.

Other Manus Administrator Settings affecting NRCS Status Feedback

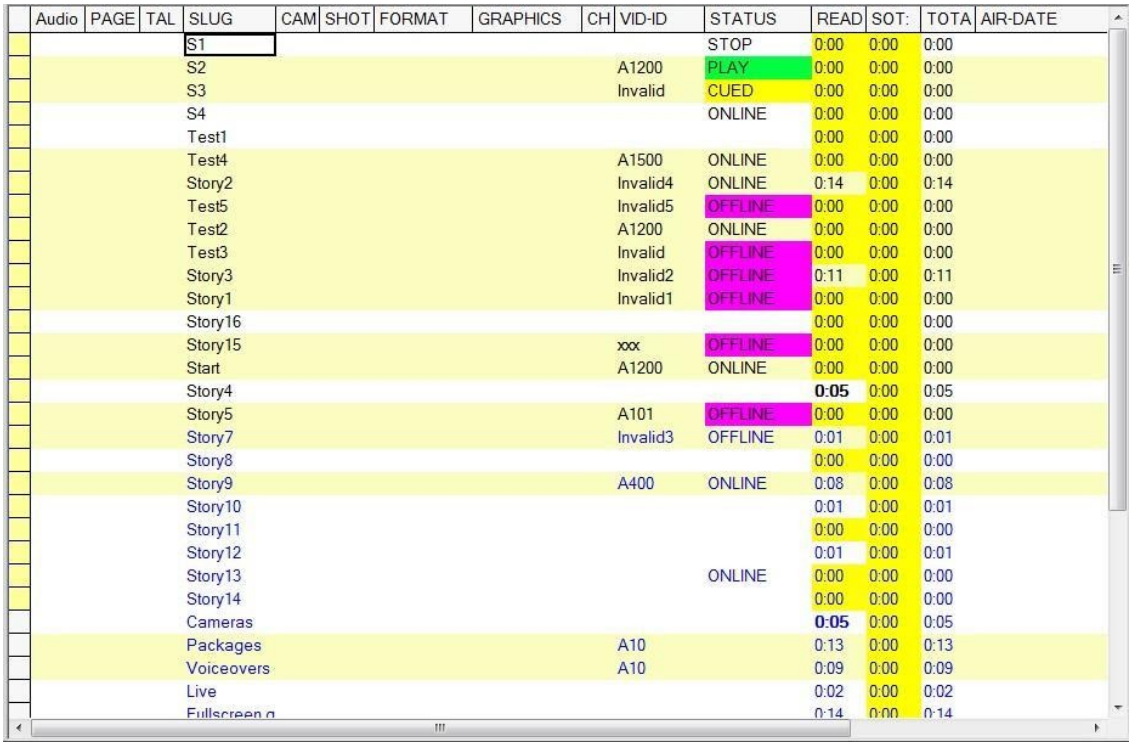
- **Send Clip Status To NRCS** (Default: false)
  Set to true to send clip status (Online/Offline) when Media Administrator is running.
- **Send only availability status for clips to NRCS** (Default: false)
  When enabled, this setting will restrict the statuses sent to NRCS to only the availability statuses for clips, specifically 'Online' and 'Offline'. Playback statuses such as 'Cue', 'Play', and 'Stop' will not be sent.
  If the rundown is in *rehearsal mode*, Viz Mosart will only send availability statuses for clips, regardless of how this setting is configured.
- **Ignore Send Cue Status to NRCS for Offline Clips** (Default: false)
  Select to prevent the NRCS receiving CUED or READY statuses of offline video clips for the first story set as next when the timeline is not running.
- **Ignore Updates If No Changes** (Default: false)
  If enabled, stories with no relevant changes will not be processed by Viz Mosart. This is to avoid issues with story updates from the Mosart GUI being ignored. Later changes in Viz Mosart iNEWS Web Services API prevented such problems, but for future changes is good to keep in mind this setting together with **StoryCompareIgnoreAttributes** which is explained below.
- **Story Compare Ignore Attributes** (Default: empty)
  List of xml attributes to ignore during comparison. Internally, the default list of attributes is set to [date_0, date_1, timestamp] and what is configured in this setting will be added to this list.

  When Viz Mosart receives a story in NSML format, it will parse the story and create its own XML format. When irrelevant updates come from NRCS, some attributes in the Viz Mosart XML story are updated. Currently these are date_0, date_1 and timestamp (see the format of a Viz Mosart XML story in any xml file in C:\manus folder). Attribute date_0 was recently added to the default attributes to be ignored due to the problems with updates caused by Viz Mosart iNEWS Web Services.
  In future if other attributes cause such issues, then they can be added to the list of attributes to be ignored.
- **Refresh Media On NRCS Update** (Default: false)
  If enabled, it forces updating clips when story is changed in NRCS. This is an old setting introduced before Viz Mosart iNEWS Web Services.
- **Keep Story Status On NRCS Update** (Default: false)
  If enabled, will keep the story status on an NRCS update, thus keeping the on-air status. The content of the story will be updated, but the story will be gray in the Mosart GUI. This is also an old setting introduced before Viz Mosart iNEWS Web Services.

Configurations in AvidInewsWebServiceConfig.xml

Copy the configuration file from in *Mosart Installation folder\Mosart Server\ConfigurationFiles* to the *ChannelTemplates* folder on C:\.

- **SendUpdateStatusForAllItems** (Default: false)
  By default, only status of items containing clips will be sent (Online/Offline/Cued/Play/Stop).
  Not only clips given through Package or Voiceover are taken into consideration, but also Adlibpix or other ways to insert clip into a story. So for example if a story contains a Camera and an Adlibpix, than the story gets status.
- **StatusColId** (Default: event-status) The id of the column to display the story item status sent by Viz Mosart.
- **AirDateColId** (Default: air-date) The id of the column to display the air-date sent by Viz Mosart.
- **ClipDurationColId** (default: empty) The id of the column to display the clip duration sent by Viz Mosart.
- **ClipDurTcFormat** This setting determines the timecode format used for displaying clip durations in the column specified by "ClipDurationColId". If the iNEWS column is not of type 'duration', the default format is 'mm:ss'. Other available formats include: 'ff', 'ss', 'hh:mm:ss:ff', 'mm:ss:ff', and 'ss:ff'. When the iNEWS column is of type 'duration', Mosart will transmit the duration in seconds. Conversion from frames to the selected timecode format is carried out based on the frame rate configured in the Manus Administrator settings.
- **ClearStatusWhenReload** (Default: false)
  If enabled, only status column is cleared on reload of the rundown. This is to avoid inconsistent status if, for example, stories failed to be updated because of a crash of Viz Mosart or iNEWS or Viz Mosart is closed while rundown is playing etc.
- **ClearStatusWhenUnload** (Default: false)
  If enabled, the status will be cleared when rundown is unloaded.
- **SetStoryAirDate** (Default: true)
  Set to false to disable setting the AIR-DATE field and moving the "aired line" (the blue line). Note that the color of the line can be changed from Tools/Options/Colors in the iNEWS client and choose “Aired” from drop-down box. By default this is blue.
- **ClearAirDateWhenReload** (Default: false)
  If enabled, the air-date field will be cleared during reload of the rundown (empty status). Note that this setting is taken into consideration only if **SetStoryAirDate** is true.
- **ClearAirDateWhenUnload** (Default: false)
  If enabled, the air-date field will be cleared during unload of the rundown (empty status). Note that this setting is taken into consideration only if **SetStoryAirDate** is true.
- **Color index attributes**
  The following parameters are colour index referring to values defined in SYSTEM.COLORS on the iNEWS system.
  The AIR-DATE field needs to be empty in order to see the correct color for the STATUS field:
  - **UrgencyIndex** (Default 1)
  - **UrgencyIndexONLINE** (Default 1) Set color for Online status.
  - **UrgencyIndexOFFLINE** (Default 4 - purple) Set color for Offline status.
  - **UrgencyIndexCUED** (Default 2 - green) Set color for Cued status.
  - **UrgencyIndexPLAY** (Default 3 - yellow) Set color for Play status.
  - **UrgencyIndexSTOP** (Default 1) Set color for Stop status.



Audio	PAGE	TAL	SLUG	CAM	SHOT	FORMAT	GRAPHICS	CH	VID-ID	STATUS	READ	SOT:	TOTA	AIR-DATE
			S1							STOP	0:00	0:00	0:00	
			S2						A1200	PLAY	0:00	0:00	0:00	
			S3						Invalid	CUED	0:00	0:00	0:00	
			S4							ONLINE	0:00	0:00	0:00	
			Test1								0:00	0:00	0:00	
			Test4						A1500	ONLINE	0:00	0:00	0:00	
			Story2						Invalid4	ONLINE	0:14	0:00	0:14	
			Test5						Invalid5	OFFLINE	0:00	0:00	0:00	
			Test2						A1200	ONLINE	0:00	0:00	0:00	
			Test3						Invalid	OFFLINE	0:00	0:00	0:00	
			Story3						Invalid2	OFFLINE	0:11	0:00	0:11	
			Story1						Invalid1	OFFLINE	0:00	0:00	0:00	
			Story16								0:00	0:00	0:00	
			Story15						xxx	OFFLINE	0:00	0:00	0:00	
			Start						A1200	ONLINE	0:00	0:00	0:00	
			Story4								**0:05**	0:00	0:05	
			Story5						A101	OFFLINE	0:00	0:00	0:00	
			Story7						Invalid3	OFFLINE	0:01	0:00	0:01	
			Story8								0:00	0:00	0:00	
			Story9						A400	ONLINE	0:08	0:00	0:08	
			Story10								0:01	0:00	0:01	
			Story11								0:00	0:00	0:00	
			Story12								0:01	0:00	0:01	
			Story13							ONLINE	0:00	0:00	0:00	
			Story14								0:00	0:00	0:00	
			Cameras								**0:05**	0:00	0:05	
			Packages						A10		0:13	0:00	0:13	
			Voiceovers						A10		0:09	0:00	0:09	
			Live								0:02	0:00	0:02	
			Fullscreen g								0:14	0:00	0:14	

- **LockFormSectionOnly** (Default: true)
  If true will issue a lock on the FORM section of a story, if false will lock the complete story.
- **HeartbeatInterval** (Default: 3 seconds)
  Interval to check Avid Web Service heartbeat.
- **UpdateStoriesCacheInterval** (Default: 1 second)
- **RetryStoryUpdateStatusInterval** (Default: 5 seconds)
- When a story failed to be updated (either because the story is locked for editing, the WebService is down or the iNEWS server is down), an error message will be displayed in console.
  The retrial to update the story status will continue until the problem is fixed or Manus Admin is closed.
- **NoOfStoriesToGet** (Default: 50) – number of stories to get from iNEWS for caching

Operation

When the connection to the Avid Web Service is successful, the following message will appear in Manus Admin console:
<iNews Web Services <host:port> connected>.
The connection will be made after loading a rundown in the GUI.

```
Viz Mosart - Manus Administrator | v5.4.0.33831 Development | IDLE - SHOW.LAURA.VR_DEMO
12:30:43 16 SITE FORMAT=2nsml.
12:30:43 16 Inews ftpserver (bgo-in8-a using FTPS, active mode, ports: random) connected: NEWS Rxnet (ftp server) 2022.1
2 (build 8.2.0.23) RH7 ready.
12:30:43 16 NCSStatus set iNews bgo-in8-a True
12:30:44 16 ChannelTemplates loaded, current studiosetup is Funcspec
12:30:44 16 Loading AvConfig
12:30:44 21 <iNews Web Services <bgo-mos-server1:18080> connected.>
12:30:44 16 Purging 0 clips from media cache)
12:30:44 26 Resyncing rundown, current show: , new show: SHOW.LAURA.VR_DEMO
12:30:44 20 SITE FORMAT=2nsml.
12:30:44 20 Inews ftpserver (bgo-in8-a using FTPS, active mode, ports: random) connected: NEWS Rxnet (ftp server) 2022.1
2 (build 8.2.0.23) RH7 ready.
12:30:44 20 NCSStatus set iNews bgo-in8-a True
ftp download time 46
ftp download time 46
ftp download time 47
ftp download time 46
ftp download time 48
ftp download time 49
ftp download time 46
ftp download time 47
ftp download time 45
12:30:45 16 Did not Initialize devices. Server in IDLE mode
ftp download time 48
ftp download time 46
12:30:45 16 Story External updated (Cam + Arc offset 30).
12:30:45 16 Story External updated (Story - Cam + Arc + Walls).
12:30:45 16 Story External updated (Story - Cam + Lwd).
12:30:45 16 UpdateClipTimings - for clip=VizrtTest1, mIn= 0, mOut=1301, dur=1301, pdur= -1.
12:30:45 16 Story External updated (Clip).
```

The following statuses will be shown in iNEWS:
Cued, *Play*, *Stop*, *Online* and *Offline*.

Info: *Offline* status also will be assigned for a clip that is to be cued but will fails cuing.

Users can move stories in iNEWS up or down. If rundown is on-air and user has been changed a position of On-Air or Next stories statuses of clips will also be changed according to status.

Note: If an iNEWS story with status OFFLINE due to "cue failed" is moved in the iNEWS playlist (even if it will be the Next story), it will be still OFFLINE and the status should stay as OFFLINE.
Check iNEWS server settings (MCS dictionary located at /site/dict/mcs) to translate the statuses shown in the iNEWS client to local statuses.

Info: No status is sent for overlay graphics.

Operational Status

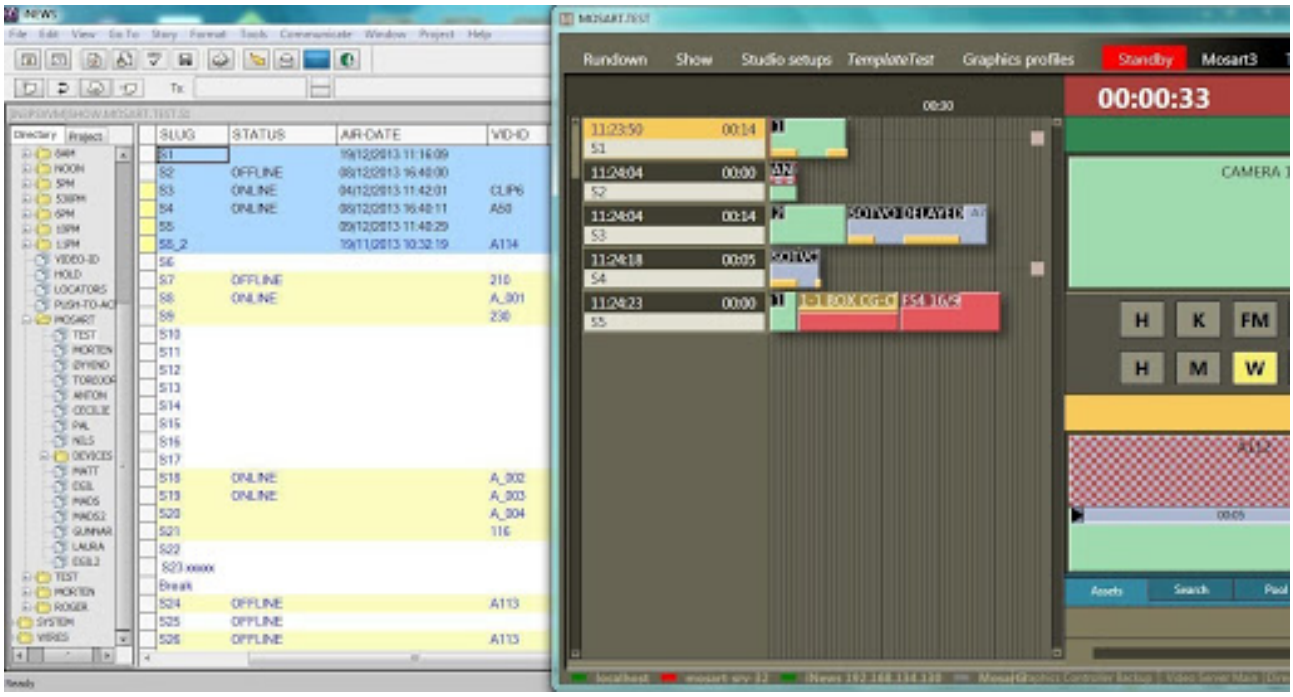
The table below shows examples of status behavior for setting **SendUpdateStatusForAllItems**.

- If **SendUpdateStatusForAllItems** is *false*, the entire column "*Story without video clips*" will be cleared (no status sent for these stories).
- **ClearStatusWhenReload** should be set to *true* to clean up old history.
- The last row (*Story has been On air*) means that after the story has been run but the rundown is still on-air, the clip is no longer monitored.
  If the clip subsequently becomes available or unavailable, this is *not* reflected in iNEWS.
  Only after the rundown is reloaded, is clip status updated.

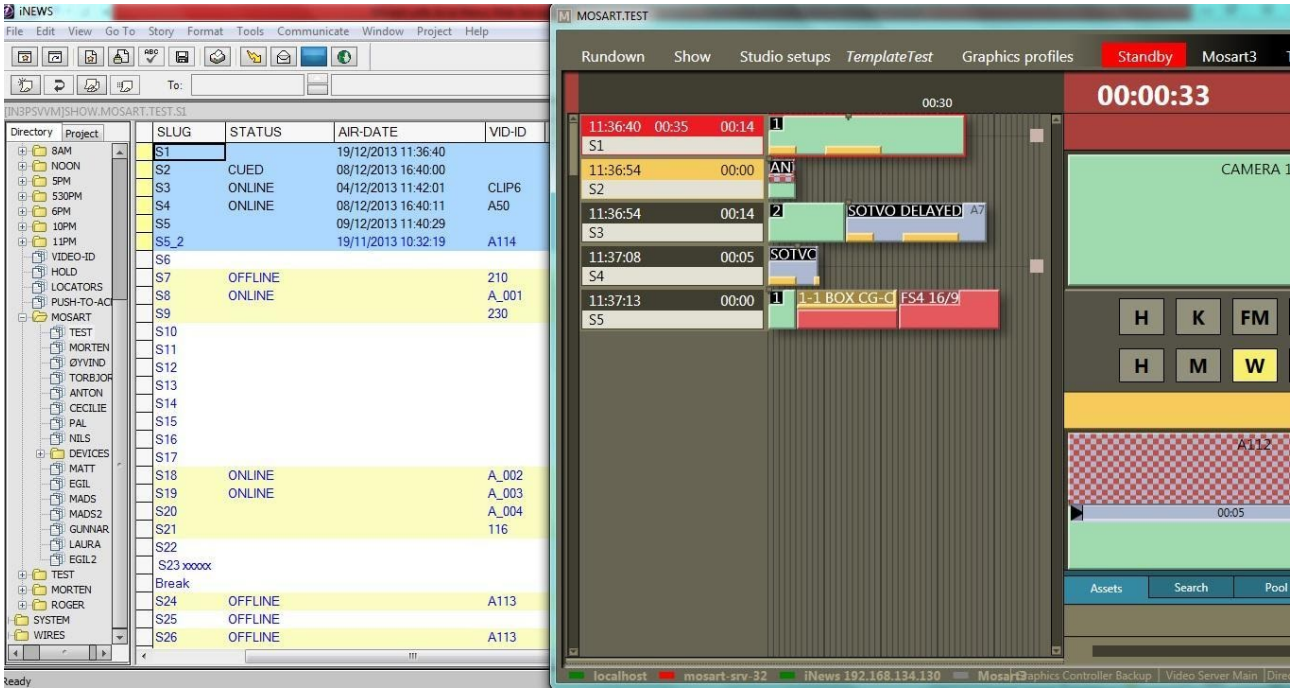
Time sequence	Story with video clips	Story without video clips (SendUpdateStatusForAllItems = *true*)	Story without video clips (SendUpdateStatusForAllItems = *false*)
Before On air	ONLINE/OFFLINE		
Next story	CUED	CUED	
Story On air	PLAY	PLAY	
Story just taken Off air	STOP	STOP	
Story has been On air	STOP	STOP	

- **Air-date** is also updated to the NRCS if the parameter **SetStoryAirDate** is set to *true* in the AvidInewsWebServiceConfig.xml file.
  When the parameter for **SendUpdateStatusForAllItems** is false, only **air-date** for stories that do not contain any clips will be updated.

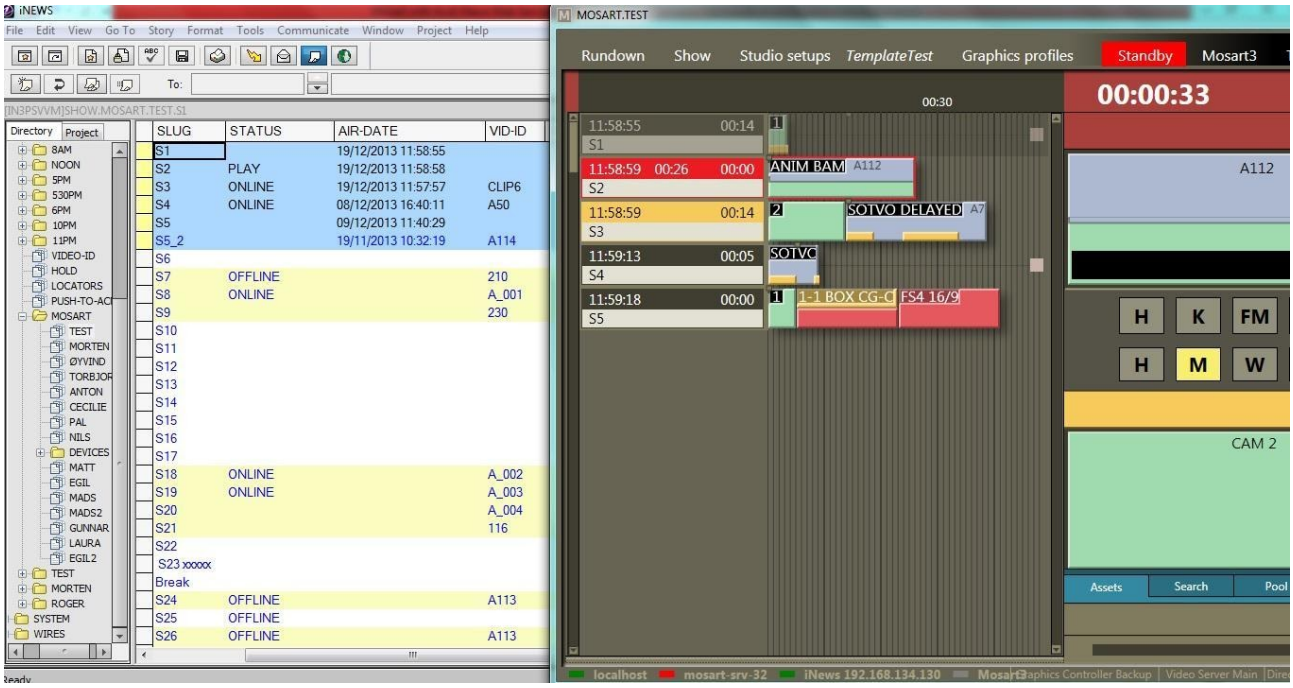
Below are examples of status behavior:



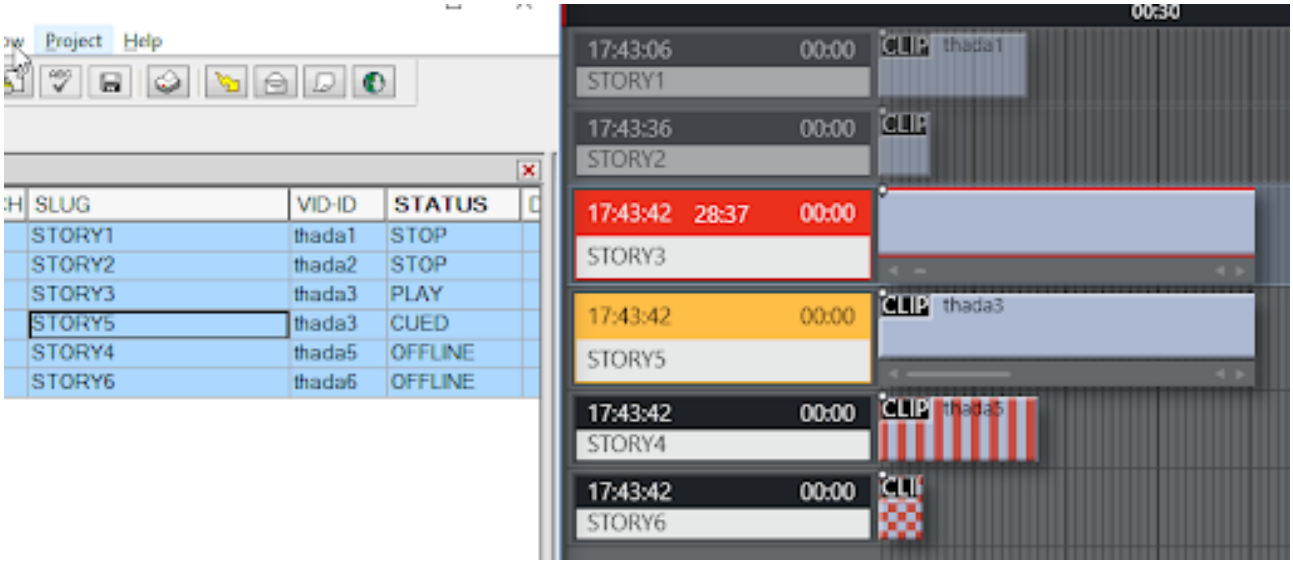
Above, the rundown is off-air. Stories S2, S3 and S4 were updated with status of the clip it contains.



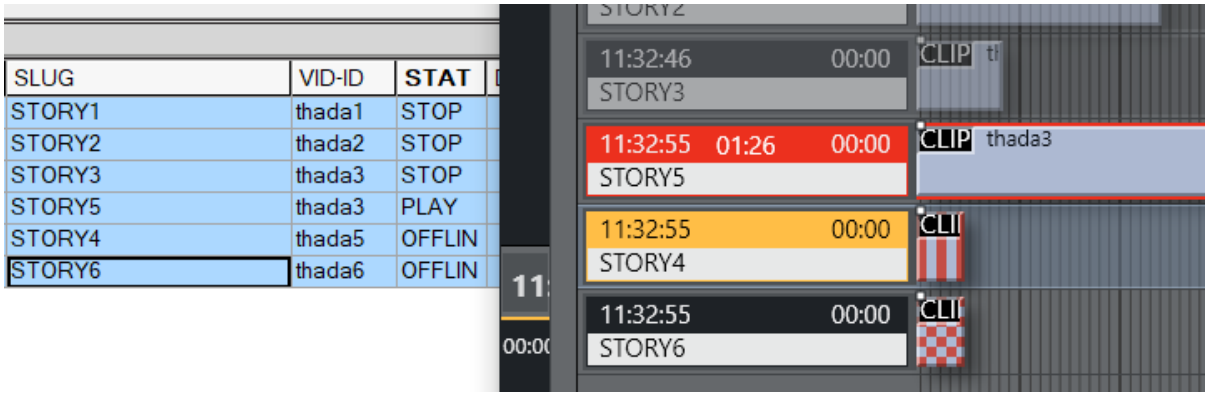
Above, the rundown is on-air. Story S1 is currently playing, but the status was not sent to the NRCS. Since S2 contains a clip, its status is set to “CUED”.



In the illustration above, Story S2 is playing and S3 is “CUED”. Since the first item in S3 is a camera the “CUED” status is not sent unless **SendUpdateStatusForAllItems** is set to *true*.



Above, Story 3 is now playing and Story 5 is “CUED”. Clip thada5 for Story 4 is broken and will be failed during cuing

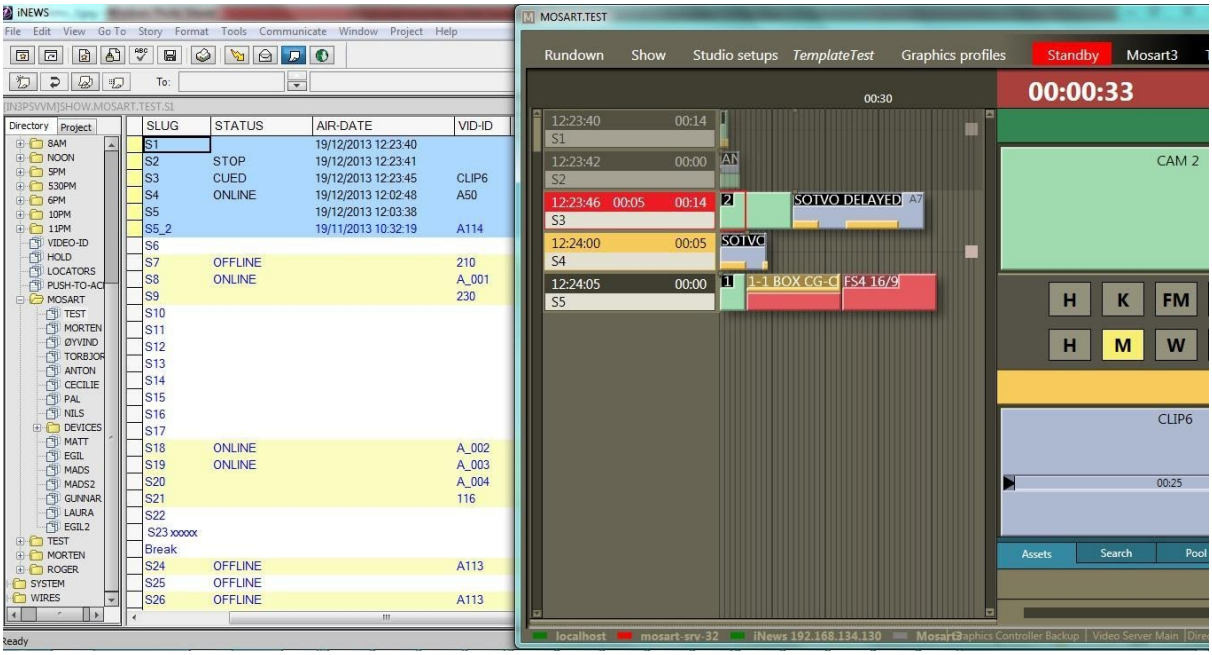


And finally, Story 5 is playing and clip from Story 4 should be cued. Zebra stripes in the Viz Mosart GUI indicate an issue with availability and Story 4 is marked as OFFLINE in iNEWS

Status for Stories Containing more than One Item

If a story has more than one item, the following statuses are sent when the story is on-air (keeping in mind that if **SendUpdateStatusForAllItems** is *false* the status is not sent if the story does not contain clips):

- PLAY for first item, CUED for next item. This means that the story status will be CUED (regardless of **SendUpdateStatusForAllItems** value)

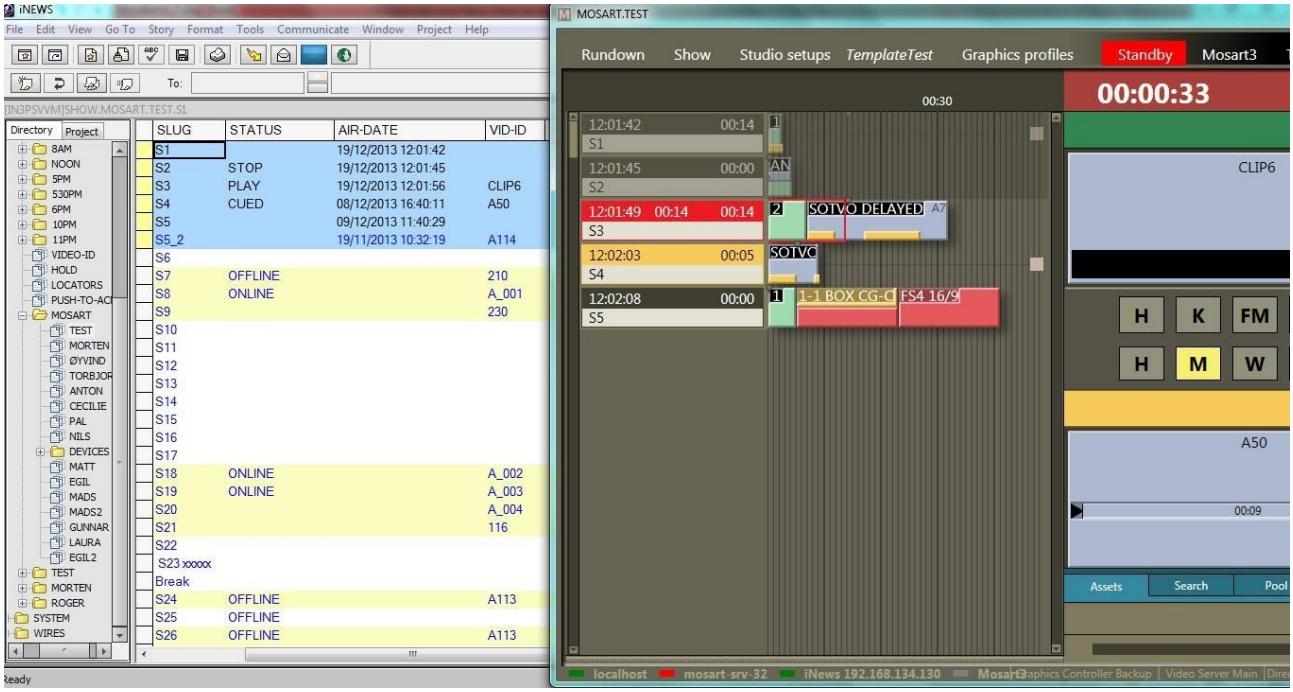


Above illustrates that when the first item (camera) in S3 is playing, no “PLAY” status is sent. The Next item is a Package so “CUED” status is sent. In this case, even if **SendUpdateStatusForAllItems** is *true*, the “CUED” status will still be sent for the story.

- When F12 is pressed, STOP is sent for the first item, PLAY for the second item and CUED for the third item. This means that the story ends up with CUED status (regardless of **SendUpdateStatusForAllItems** value). This continues until we reach the last item in the story.
- It is only the last item that will send the status PLAY. At any time, it is possible to check the AIR-DATE to identify the story currently on-air.

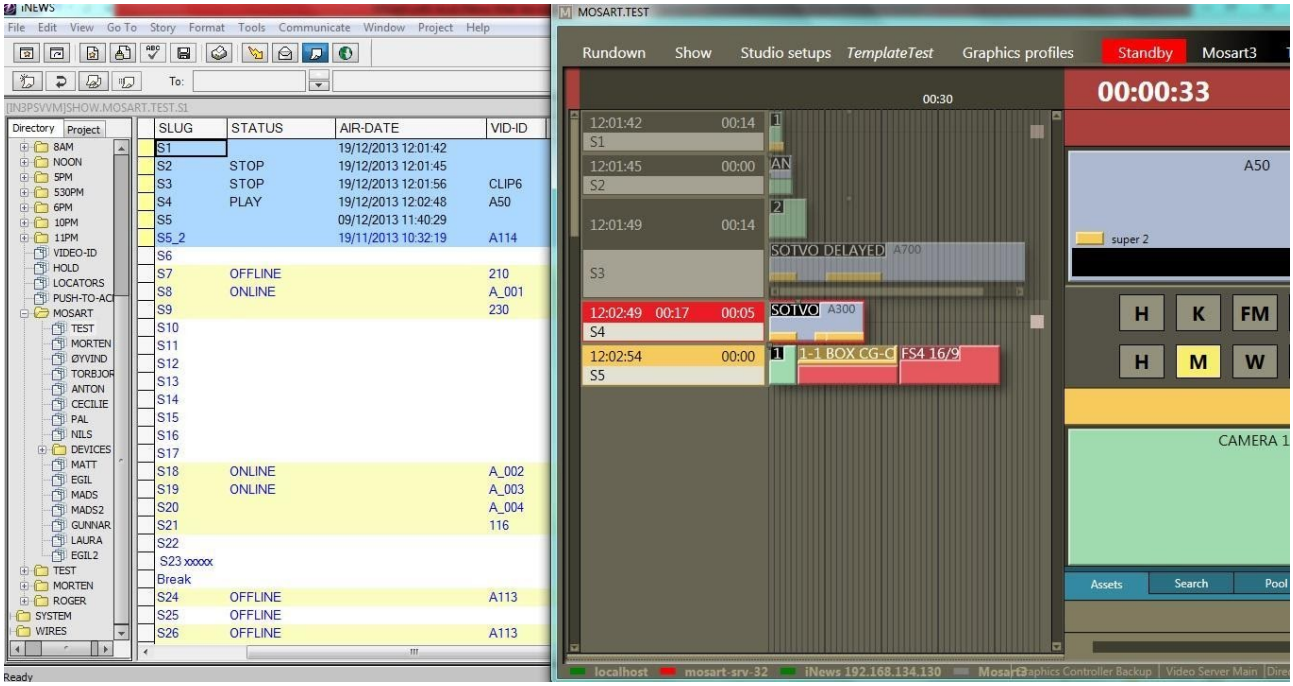
Continuing this example (2 screenshots below), the story S3 has only two items, a Camera and a Package. When we play the clip in the Package, the story gets PLAY status.

The Package from S3 is playing, this is the last item in the story so “PLAY” status is set. Story S4 will get “CUED” because it contains a clip.



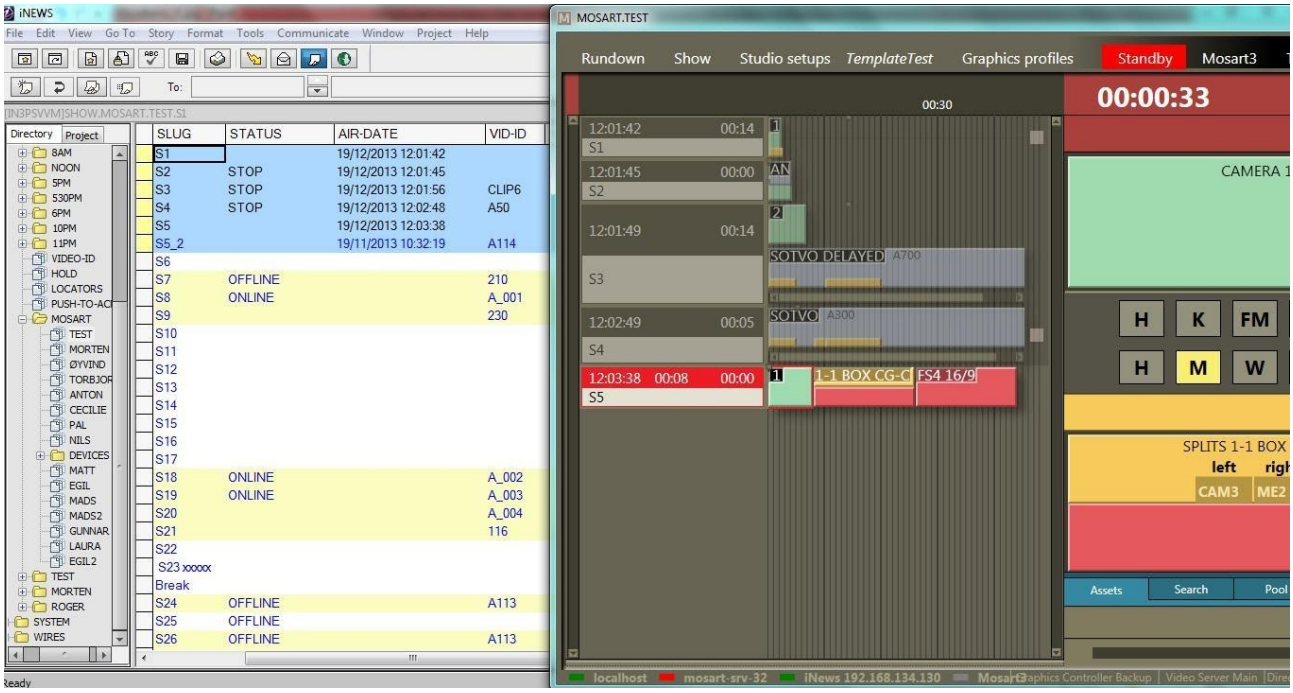
Example 1

Story S4 containing only a clip, is playing. No “CUED” status has been sent for next story since **SendUpdateStatusForAllItems** is set to *false*.



Example 2

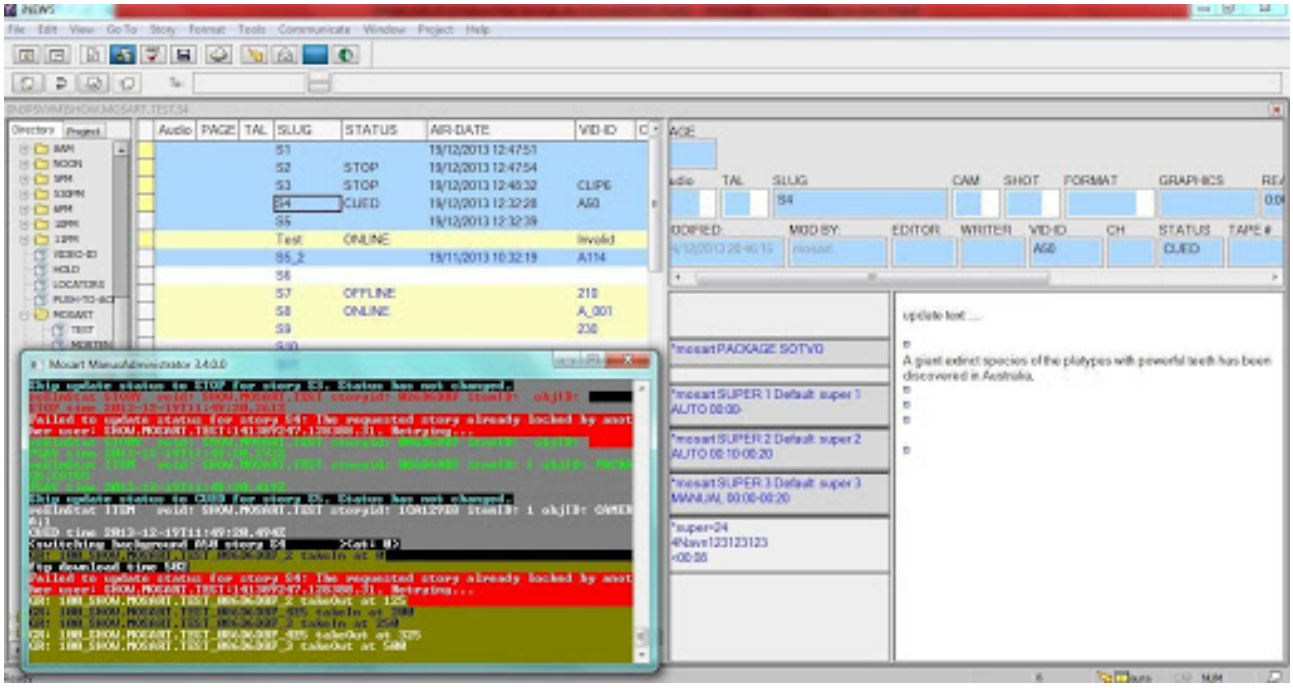
Story S5 is playing and since the story does not contain any clips, no status will be seen for this unless **SendUpdateStatusForAllItems** is set to *true*.
Only the “AIR-DATE” field shows that this story is on-air.



Example 3

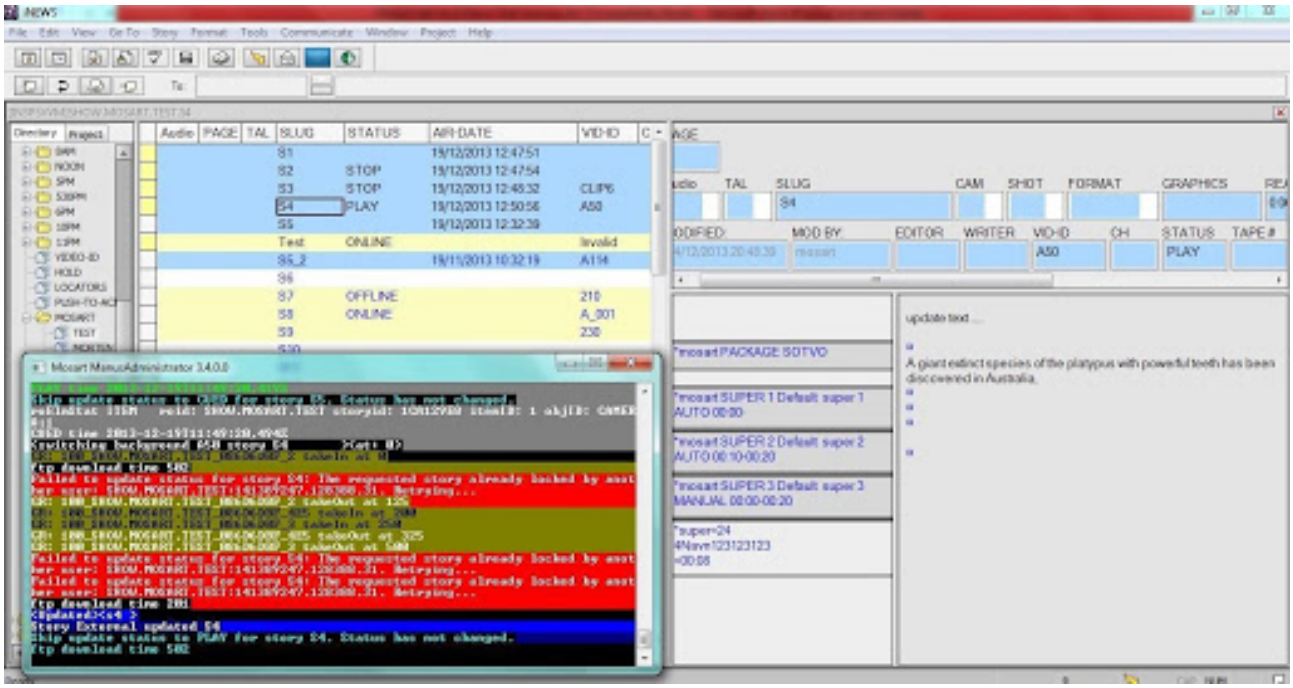
Play status could not be sent for story S4.

When Viz Mosart is performing the updates, it locks the form of the story or the entire story depending on the value for **LockFormSectionOnly** parameter and unlocks the story after the update is performed.
If the story form is locked by an iNEWS user, a message in the Manus Admin console warns the user that the status could not be sent because the story form is locked. Once the story form is unlocked, the status is updated (see Example 4).



Example 4

Story S4 is now unlocked in iNEWS and “PLAY” status has been sent.



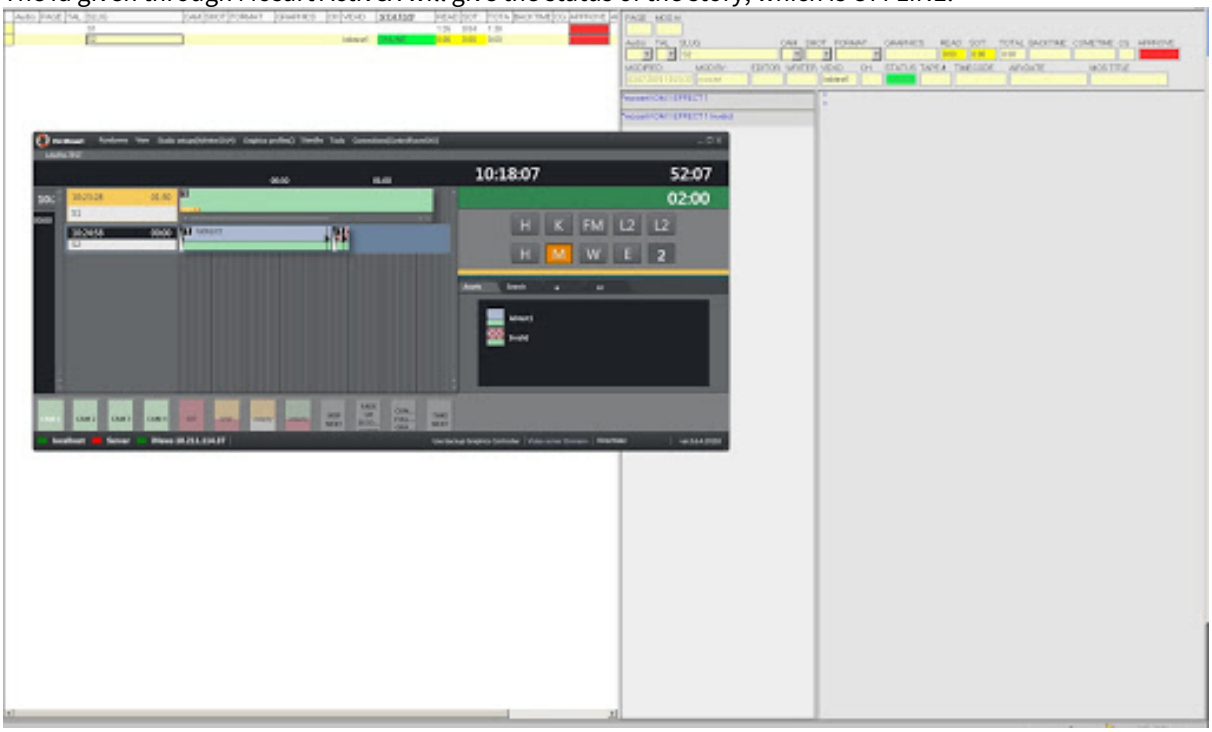
If the timeline is not running and the first story is floated, you may have to reload the rundown in the Mosart GUI in order to set CUED status for the new first story.

Status of an Off-air Story Containing Multiple Clips

The status of an off-air story that contains more than one clip is provided by the first clip in the story.

Example:

1. A story with VID-ID set and containing two clip items.
2. First clip item has id given by the VID-ID field and it's an Online clip.
3. Second clip item is given an id, for example, through clip_hirespath attribute (Mosart ActiveX) and it is an Offline clip.
4. The story will have the status depending on the availability of the first clip, which is OFFLINE.
5. If the position of the clips is changed, then the first clip will get 2 IDs, the one from VID-ID and the one given through Mosart ActiveX.
6. The id given through Mosart ActiveX will give the status of the story, which is OFFLINE.

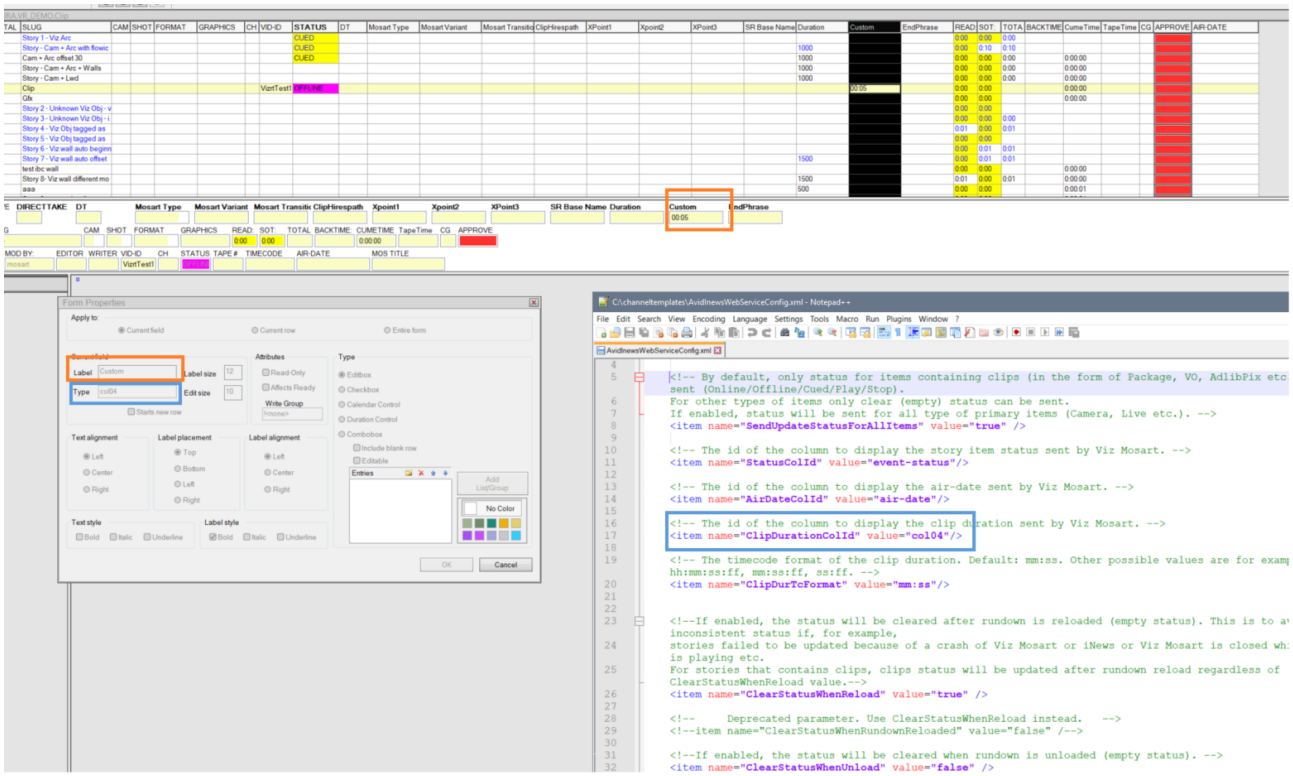


Story *S2* gets the status of the first clip.

Clip Duration Feedback to iNEWS Server

- The clip duration sent to iNEWS will be in mm:ss timecode format by default. Other formats are possible to configure. A system-wide frame rate for conversion is assumed and can be configured in Mosart Manus Admin settings.
- iNEWS can only display information at story level and therefore rules have been defined when a story contains more than one clip. However, for optimal usage of this functionality, it is *recommended to have stories containing only one clip*.
- A column must be configured in iNEWS to show the duration of a clip found in the story. A column id must also be configured in Viz Mosart, in the file *AvidInewsWebServiceConfig.xml*.

For status and air date, the default iNEWS columns have been used, but from Mosart 5.1.1, any column can be configured to show the status or the airdate.



- When a rundown is loaded/reloaded in Mosart, Mosart will update all the stories in iNEWS with *the duration of the first clip* found in each story, in addition to the already existing clip availability status.
- When a clip is offline, the default duration configured in Mosart will be sent.
- When a clip update is received in Mosart, Mosart will send the updated clip duration to iNEWS.
- In general, when a given story item is cued/played/stopped, Mosart will send the clip duration if that item is a clip.
- If the timeline is not running and the item following the current item cued is a clip, the story will be updated with CUED status and the duration of the clip.
- If a clip that is currently being recorded or copied to the MAM system is included in the rundown, its duration is automatically transmitted to iNEWS, assuming that Mosart supports the 'growing clip' functionality for the configured video server.

Rules

- If a story has multiple items, the following rules apply (this is not new, but the current functionality has impact on how the clip duration is displayed):
    - When the *story is cued*, the status CUED and the duration of the first item in the story, if this a clip, will be sent to iNEWS.

- When the *first item in the story is taken*, Mosart sends status STOP for the previous item and the duration if this was a clip.
- Mosart also sends PLAY *for the first item in the story* and CUED *for the next item in the same story*. This means that the story ends up with status CUED and the clip duration of the cued item if this is a clip.
- The same logic is applied until the last item in the story is taken.
- When the *last item in the story is taken*, Mosart sends PLAY status and the clip duration if this is a clip.

Example:

In the following rundown:

Story S1: Clip1
Story S2: Cam1 Cam2 Clip2 Clip3 Cam3
Story S3: Cam Clip4
Story S4: Cam

The following settings are configured:

- **StatusColId**, **AirDateColId** and **ClipDurationColId***:* Set to the corresponding iNEWS column id.
- **SendUpdateStatusForAllItem:** *true*
- **SetStoryAirDate***: true*
- **Send Clip Status To NRCS**: *true*

When the rundown is loaded, the following information is sent to iNEWS:

Action	Description	Status (iNEWS)	Duration (iNEWS)	Air-Date (iNEWS)
S1 is cued		CUED	Clip1 duration	
S2		ONLINE/ OFFLINE	Clip2 duration	
S3		ONLINE/ OFFLINE	Clip4 duration	
S4				

When the rundown is played, the following information displayed in the iNEWS columns:

Action	Description	Status (iNEWS)	Duration (iNEWS)	Air-Date (iNEWS)
S1 played		S1 PLAY S2 CUED	S1 Clip1 duration	Updated
S2 Cam1 taken	Cam1 is played and Cam2 is cued on S2.	S1 STOP S2 CUED	S1 Clip1 duration	Updated

S2 CAM2 is taken	Cam1 is stopped, Cam2 is played and Clip2 is cued.	S2 CUED	S2 Clip2 duration	
S2 Clip2 is taken	Cam2 is stopped, Clip2 is played, Clip3 is cued.	S2 CUED	S2 Clip3 duration	
S2 Clip3 is taken	Clip2 is stopped, Clip3 is played, Cam3 is cued.	S2 CUED		
S2 CAM3 is taken	Clip3 is stopped, Cam3 is played, S3 Cam is cued.	S2 PLAY S3 CUED		
S3 CAM is taken	S2 Cam3 is stopped, S3 Cam is played, S3 Clip4 is cued.	S2 STOP S3 CUED	S3 Clip4 duration	Updated
S3 Clip4 is taken	S3 Cam is stopped, Clip4 is played, and S4 Cam is cued.	S3 PLAY S4 CUED	S3 Clip4 duration	
S4 Cam is taken	S3 Clip4 is stopped, S4 Cam is played S4 Cam is cued	S3 STOP S4 CUED	S3 Clip4 duration	Updated

Troubleshooting and Known Issues

- Not able to Access Web Service page after Installation
- The Connection to iNEWS Server Times-out
- "No sessions available on X" – Maximum session
- "Cannot write to iNews Webservice, connect before selectQueue"
- Cannot get any status feedback in iNEWS
- The Web Service version is not allowed to connect to iNEWS Server

Not able to Access Web Service page after Installation

If you have problems accessing the Web Services page after installation and deployment (http://<server>:8080/inewswebservice/services), try to delete the Tomcat cache as follow:

1. Stop the Tomcat service.
2. Delete folder Tomcat installation\webapps\inewswebservice (only the folder, not the *war* file).
   You may replace the war file, for example, with a new one when upgrading (this solution is same as deploying from Tomcat Manager).
3. Delete folder Tomcat installation\work\Catalina\localhost\inewswebservice
4. Start the Tomcat service and try again to access the Web Services page.

The Connection to iNEWS Server Times-out

If the connection to the iNEWS server through the iNEWS Web Service times out, check first the iNEWS server logs for more details.

If the iNEWS server logs contain "SSL_access failed", check what version of Java you are using. SSLv3 is disabled in Java 8 by default.

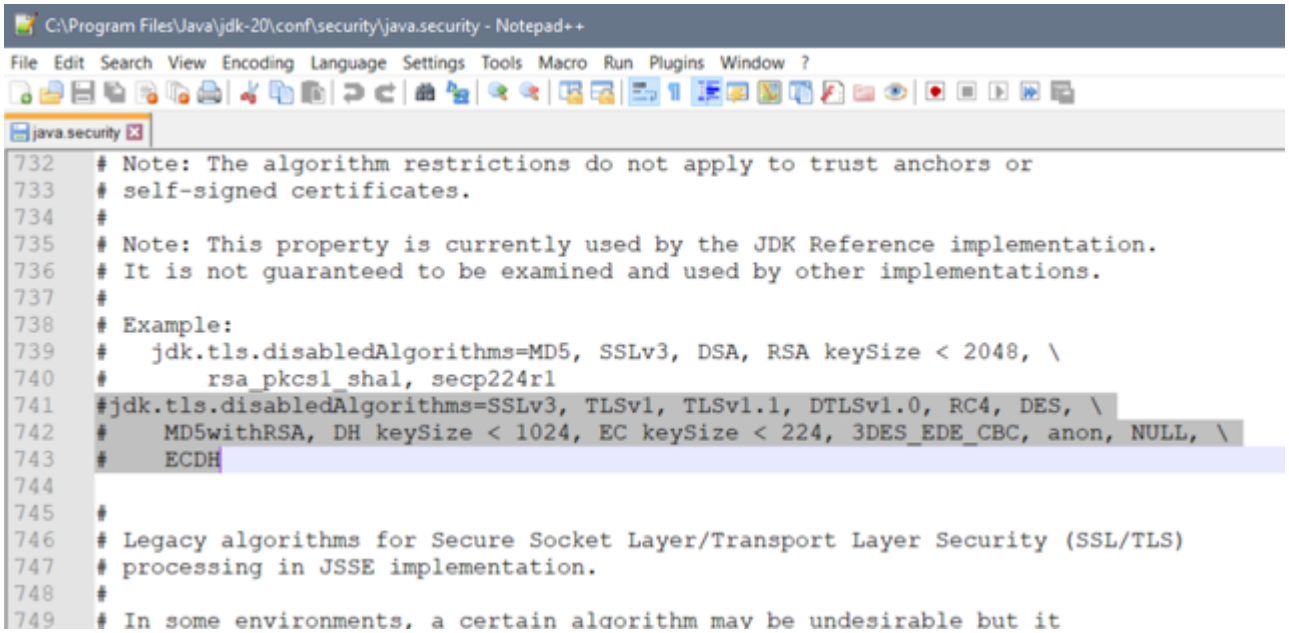
A default Java 8 installation disables the SSLv3 algorithm, preventing WSAPI from connecting to the Newsroom Management Server.

- To re-enable SSLv3 algorithm, open as administrator the file java.security located in the *lib/security* or *conf\security* in your JRE or JDK installation folder

/user/lib/jvm/java-.1.8.0-openjdk-1.8.0.xxxxx/jre/lib/security

C:\Program Files\Java\jdk-20\conf\security\java.security

and comment out the lines that disables SSLv3. Restarting the Apache Tomcat service once the modifications are done.



"No sessions available on X" – Maximum session

If the above error is shown in Manus Admin's console or the log (where X is the iNEWS server IP address or hostname), refer to the section API Session Limits.
If this is not the case, then most probably the issue is that the Manus Admin console failed to release the occupied API session to connect to iNEWS server through Avid Web Services, when the API session was closed.

Workaround: There are two solutions for this problem.

1. Either restart Apache Tomcat service or
2. Reduce the session timeout in Tomcat to *1 min*, freeing-up the API session 1 min after Manus Administrator was closed.

If Manus Admin is closed and opened more than maximum API sessions allowed in 1 min, it will reconnect automatically to Web Service after 1 min when at least one session has been released.
To change the session timeout interval do the following:

a. Stop the Tomcat service.
b. Change Tomcat session timeout interval in Tomcat installation/conf/web.xml to 1 min:

```
<session-config>
    <session-timeout>1</session-timeout>
</session-config>
```

c. Start the Tomcat service.

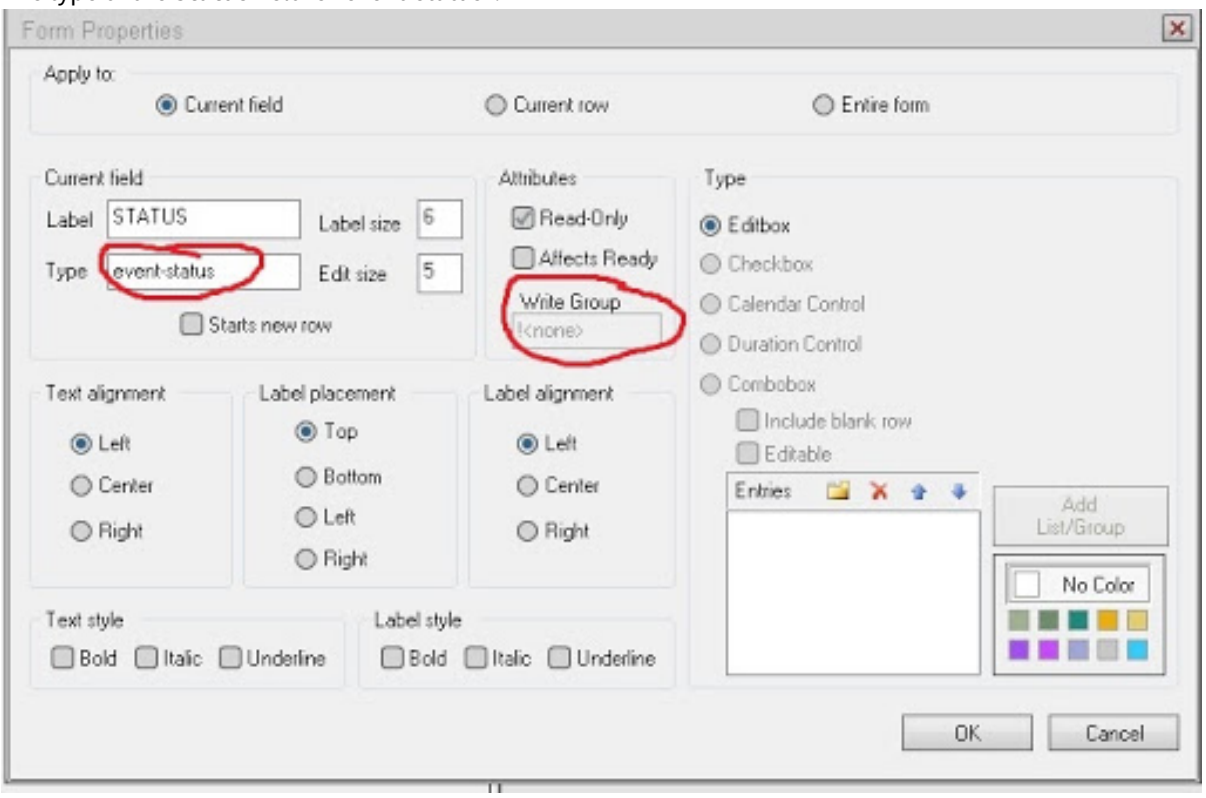
“Cannot write to iNews Webservice, connect before selectQueue”

If you get the error message “cannot write to iNews Webservice, connect before selectQueue” check that the version of Web Service running corresponds with the version set in iNEWS.
See section Checking the Current Version of Tomcat.

Cannot get any status feedback in iNEWS

If after setting up everything as described in the document, no status is written in the **Status** field, check the following:

- The **Status** field has write permissions for the user that connects Viz Mosart to the iNEWS system.
- The type of the **Status** field is “event-status”.



- The queue must *not* be monitored with the iNEWS Monitor tool.

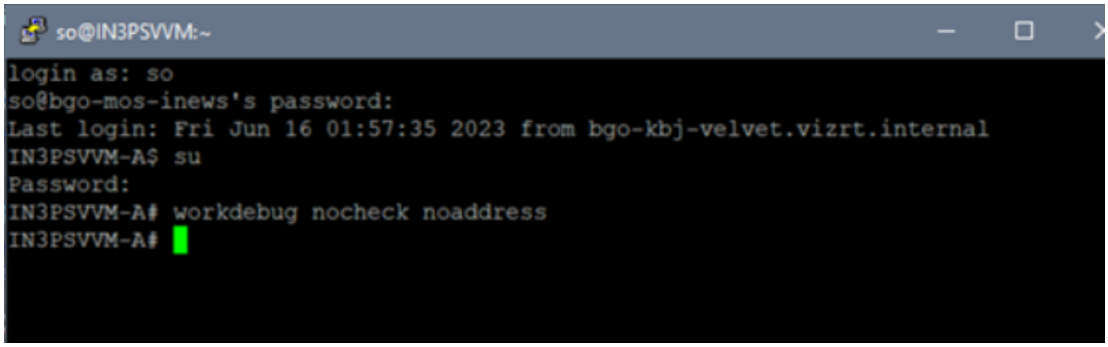
Verify also the Mosart logs or the iNEWS Web service logs located under C:\Program Files\Apache Software Foundation\Tomcat 9.0\logs\iNEWS_WS for any error related to status feedback to iNEWS.

If in the logs you encounter the error message "*The Web Service version is not allowed to connect to…*", see the next section.

The Web Service version is not allowed to connect to iNEWS Server

If you encounter the issue *"The Web Service version is not allowed to connect to…"* in the logs, disable the version check on the iNEWS server by executing `workdebug nocheck noaddress` on the iNEWS server.

This issue is common with Mosart version 5.4.0 and above, where the iNEWS Web Service has been upgraded to version 1.6.3. Avid support recommends running this command after every iNEWS server reboot, following the standard connect and startup commands.



Useful Links

- http://tomcat.apache.org/whichversion.html
- http://java.com/en/download/index.jsp
- iNEWS Mos Gateway Installation and Operations Manual v2.6.0
- iNEWS Mos Gateway v4.0 ReadMe
- iNEWS v4.5 Setup and Configuration Guide
- Introduction to iNEWS Workstation

Copyright © 2026 Vizrt

12.1.5 Jupiter Systems PixelNet Video Wall Control

This section covers the following topics:

- Introduction
- Configuration
  - AVAutomation
  - Template Editor
  - Configuration File
- Implementation
  - Connection
- Requests to Implement
  - To use LoadLayoutEx
  - To use ActivateWall and LoadLayout
  - To use LoadLayoutOnWall
  - To use ActivateWall with LoadLayoutOnWall
- AVAutomation Implementation

Introduction

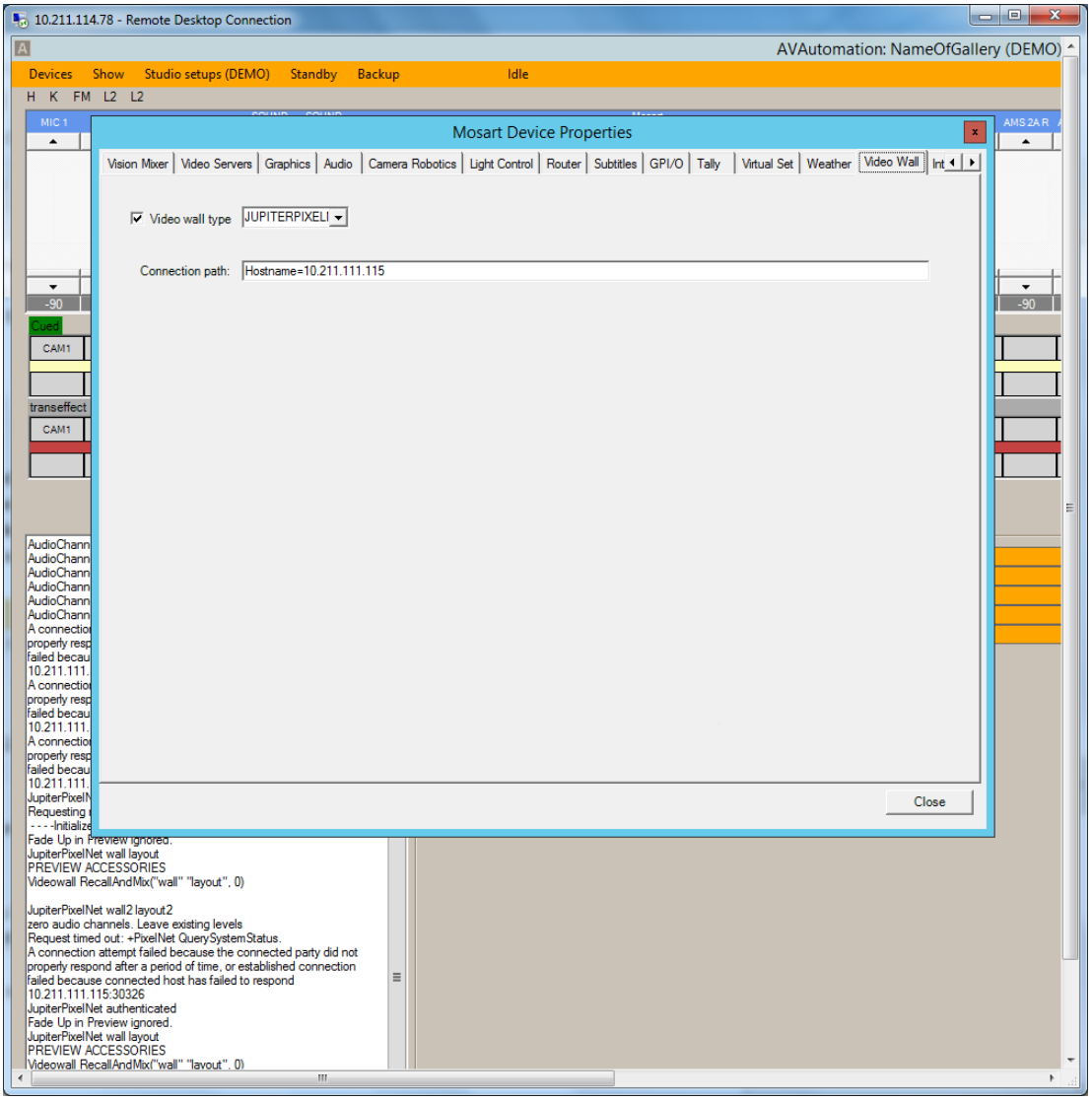
The protocol is text based commands and responses sent on TCP/IP. Text encoding is UTF-8. The protocol is synchronous - i.e. a command (request) shall always result in a response, and a new command shall not be sent until the response to the previous command has been received.

Configuration

AVAutomation

The router protocol is named JUPITERPIXELNET. Configuring the PixelNet video wall interface requires the following parameters:

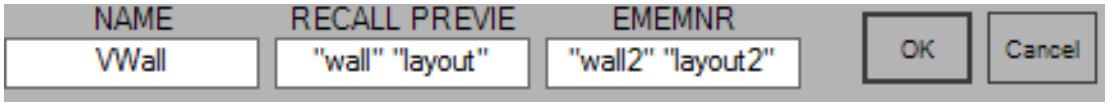
- IP address of the PixelNet Domain Controller
- Port number (default is set to 30326)



Template Editor

In the template editor, enable Video Wall. In Device Properties, enter wall and layout (in double quotes) in that order, space separated, in both ‘RECALL PREVIE’ and EMEMNR:

Implementation may be based on the WATCHOUT protocol implementation.



Configuration File

The PixelNet configuration file is located in the program-folder under *\Mosart Medialab\Mosart Server\ConfigurationFiles\VideoWallJupiterPixelNet.xml.* To use it, copy it to c:\channeltemplates, and append

;Config=VideoWallJupiterPixelNet.xml

to the connection string.

The following configurations are available:

Setting	Description	Default
PingInterval	The amount of time (in ms) to wait before ping after connect, ping or RecallAndMix	10000 (10s)
Port	The port number of the PDC computer	30326
ReconnectInterval	The amount of time (in ms) to wait before reconnect after unsuccessful connect or lost connection	10000 (10s)
Server	The DNS name of the PDC computer	None
TimeoutActivateWall	The timeout (in ms) for the ActivateWall request	2000 (2s)
TimeoutLoadLayout	The timeout (in ms) for the LoadLayout request	2000 (2s)
TimeoutLoadLayoutEx	The timeout (in ms) for the LoadLayoutEx request	2000 (2s)
TimeoutLoadLayoutOnWall	The timeout (in ms) for the LoadLayoutOnWall request	2000 (2s)
TimeoutQuerySystemStatus	The timeout (in ms) for the QuerySystemStatus request	2000 (2s)
UseActivateWall	Whether (True/False) an ActivateWall request is sent before the LoadLayoutOnWall request. Has no effect unless UseLoadLayout is False.	False
UseDispatcherForRecallAndMixes	Whether (True/False) the RecallAndMix requests are executed in the dispatcher, i.e. in the same Thread as (re-)connects and pings.	False
UseLoadLayout	Whether (True/False) LoadLayout requests are sent instead of LoadLayoutEx requests	False

UseLoadLayout OnWall	Whether (True/False) LoadLayoutOnWall requests are sent instead of LoadLayoutEx requests	False
WallIsFirstArgu ment	Whether (True/False) the Wall argument of the LoadLayoutOnWall request is the first Argument	False
WallIsInt	Whether (True/False) the Wall argument of the LoadLayoutOnWall request is an integer	False

Implementation

Connection

- PixelNet uses the PixelNet Domain Control (PDC) protocol. PDC protocol is developed on top of the Remote Method Call (RMC) inter-process communication protocol, implemented over a TCP/IP connection.
- The Watchout video wall driver may serve as a model for this driver.
- The protocol can be tested over a Telnet session.
- Port number to connect to is 30326.

Requests to Implement

Five requests are implemented:

- ActivateWall (used only if UseLoadLayout is True, or if both UseActivateWall and UseLoadLayoutOnWall are True)
- LoadLayout (used only if UseLoadLayout is True)
- LoadLayoutEx (used if both UseLoadLayout and UseLoadLayoutOnWall are False)
- LoadLayoutOnWall (used only if UseLoadLayout is False, and UseLoadLayoutOnWall is True)
- QuerySystemStatus (used for (internal) Ping)

Note: Although LoadLayoutOnWall will likely not work, it was included since version 1.5 indicated that this was the request that should be utilized.

To use LoadLayoutEx

- Leave all Use… settings at their default False.

To use ActivateWall and LoadLayout

- Set UseLoadLayout = True.

To use LoadLayoutOnWall

- Set
    - UseLoadLayout = False
    - UseLoadLayoutOnWall = True

- WallIsFirstArgument = True
- WallIsInt = True.

Note: LoadLayoutOnWall may not work.

To use ActivateWall with LoadLayoutOnWall

- UseActivateWall = True.

Note: ActivateWall with LoadLayoutOnWall may not work.

AVAutomation Implementation

The following methods declared in MMVideoWall must be implemented:

```
Public MustOverride Function Connect(ByVal strConnection As String) As String Implements IConnect.Connect
Public MustOverride Function Disconnect() As String Implements IConnect.Disconnect
Public MustOverride Function Ping() As String Implements IConnect.Ping
Public MustOverride Sub RecallAndMix(ByVal presetName As String, ByVal transrate As Integer)
```

Note: The final method in the list above, RecallAndMix, calls LoadLayoutOnWall.

12.1.6 Panasonic Kairos Vision Mixer

- General Switcher Configurations
- Specific configurations for Kairos switcher
- Crosspoints
- Effects
- Managing Macros

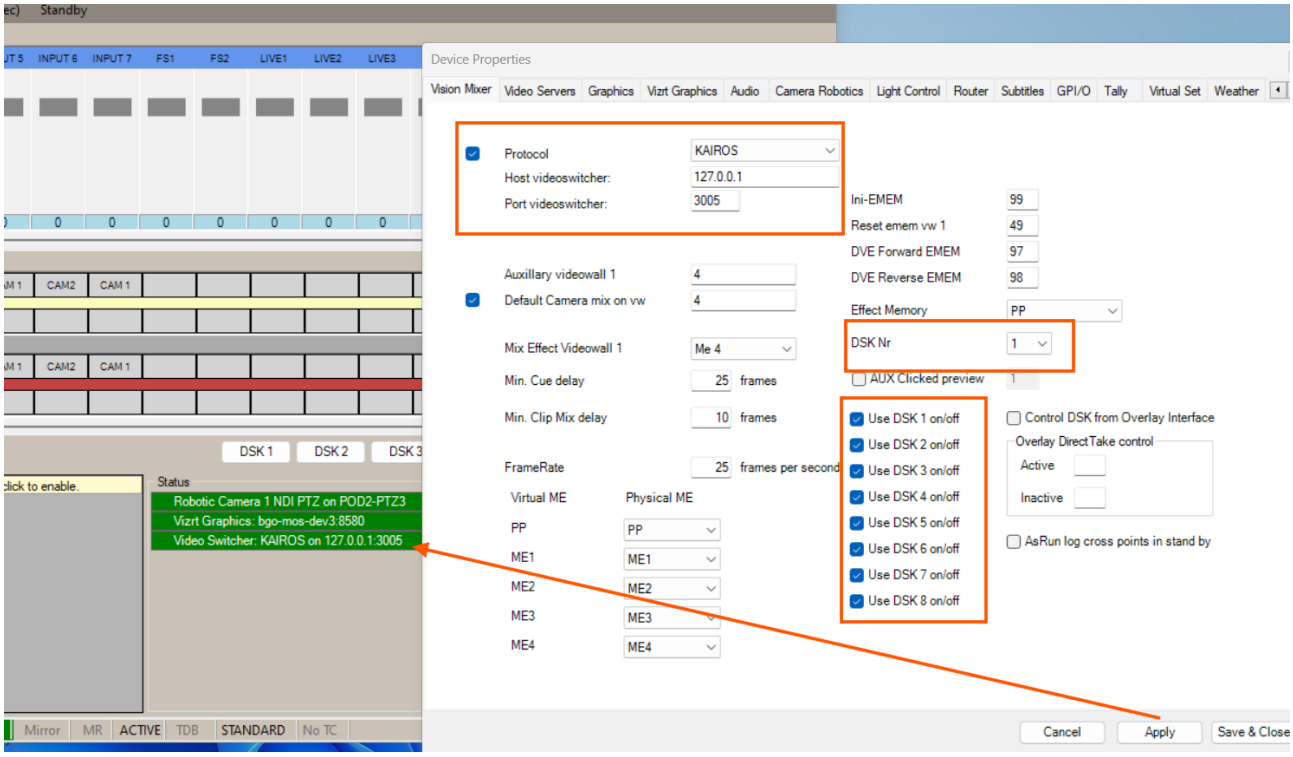
This section describes how to configure the Panasonic Kairos driver. The configuration consists of three parts:

- The general connection details and other general switcher properties in **AV Automation** menu bar **> Devices > Properties > Vision Mixer**.
- Specific Kairos configurations in the *KairosVideoSwitcherConfig.xml*.
- The configuration of the crosspoints and effects from the **AV Automation** menu bar **> Devices > A/V Setup**.

Configurations

General Switcher Configurations

From the AV Automation menu bar select **Devices > Properties > Vision Mixer** to configure the connection to the switcher and which downstream keyers (DSKs) to use.
Apply the settings and verify the connection status in the AV Automation UI.



The mapping of the main scene (PP) layers configured for DSKs to Viz Mosart's naming conventions is defined in a dedicated configuration file (*KairosVideoSwitcherConfig.xml*).
See below for details.

> **Note**: In theory, an unlimited number of DSKs can be configured, but only *8* DSKs can actually be used due to current limitations of entering values in the UI (of **AV Automation > Devices > Vision Mixer**) and Template Editor design.

Specific configurations for Kairos switcher

Some properties can be configured from **AV Automation > Devices > Vision Mixer**, but most of the settings need to be done in the *KairosVideoSwitcherConfig.xml* file located in the Mosart installation folder, typically at *C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles*.

KairosVideoSwitcherConfig.xml

Copy the file into the *C:\ProgramData\Mosart Medialab\ConfigurationFiles* and open it to edit.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="KairosVideoSwitcherConfig">
<Properties>
    <!-- Mosart sends a keep-alive message every 5 seconds (default), according to
the Simple Panasonic Kairos Control Protocol,
         to maintain an active connection. The value should be less than 10s. -->
    <item name="HeartbeatIntervalSs" value="5"/>

    <!-- The total number of mixer regions (scenes) configured on Kairos, including
the Program/Preview (PP)
         region. Default: 10. -->
    <item name="NumberOfRegions" value="10" />

    <!-- Map the scenes configured in Kairos to the naming conventions required by
Mosart.
         The Program/Preview (PP) region is mandatory.
         Each scene should be configured in Kairos to have a main layer called
Background (the name is not configurable).-->
    <item name="PP" value="Main" />
    <item name="ME1" value="Scene-1" />
    <item name="ME2" value="Scene-2" />
    <item name="ME3" value="Scene-3" />
    <item name="ME4" value="Scene-4" />
    <item name="ME5" value="Scene-5" />
    <item name="ME6" value="Scene-6" />
    <item name="ME7" value="Scene-7" />
    <item name="ME8" value="Scene-8" />

    <!-- Define the default transition as configured in Kairos.
     Note: While Kairos supports multiple transitions and complex combinations for a
single scene,
     Mosart is limited to supporting only one transition at a time for its automation
workflows.
```

```
     On Kairos, define a transition for the main scene (PP) and assign a transition
type for the Background layer.
     The name of the transition configured in Kairos should be mapped here to be used
in the commands sent to Kairos.
     Mosart will set the duration and perform the auto on this transition (not on the
"Next Transtion" which is the global auto to control all the layer transition)-->
    <item name="Transition" value="BgdMix"/>

    <!-- The number of downstream keyers (DSKs) defined in Kairos for the layers of
the Program/Preview (PP) region. These are in addition to the Background layer.
Default: 8. -->
    <item name="NumberOfDSKs" value="8"/>
    <!-- Map the layers of the Program/Preview (PP) region in Kairos to Mosart naming
conventions (DSK followed by a number). -->
    <item name="DSK1" value="Layer-1"/>
    <item name="DSK2" value="Layer-2"/>
    <item name="DSK3" value="Layer-3"/>
    <item name="DSK4" value="Layer-4"/>
    <item name="DSK5" value="Layer-5"/>
    <item name="DSK6" value="Layer-6"/>
    <item name="DSK7" value="Layer-7"/>
    <item name="DSK8" value="Layer-8"/>

    <!-- The number of key buses representing layers of mixer regions (scenes) other
than PP. Default: 8. Possible values are 4, 6, or 8. -->
    <item name="NumberOfKeyBuses" value="8"/>
    <!-- Map the layers of scenes other than the main one (e.g., ME1, ME2) in Kairos
to Mosart naming conventions (KeyFill followed by a number). -->
    <item name="KeyFill1" value="Layer-1"/>
    <item name="KeyFill2" value="Layer-2"/>
    <item name="KeyFill3" value="Layer-3"/>
    <item name="KeyFill4" value="Layer-4"/>
    <item name="KeyFill5" value="Layer-5"/>
    <item name="KeyFill6" value="Layer-6"/>
    <item name="KeyFill7" value="Layer-7"/>
    <item name="KeyFill8" value="Layer-8"/>

    <!-- The number of outputs configured in Kairos to be mapped as auxiliaries in
Mosart. Default: 16. -->
    <item name="NumberOfAuxBuses" value="16"/>
    <!-- Map auxiliary sources in Kairos to Mosart naming conventions (AUX followed
by a number). -->
    <item name="AUX1" value="IP-AUX1"/>
    <item name="AUX2" value="IP-AUX2"/>
    <item name="AUX3" value="IP-AUX3"/>
    <item name="AUX4" value="IP-AUX4"/>
    <item name="AUX5" value="IP-AUX5"/>
    <item name="AUX6" value="IP-AUX6"/>


    <!-- Map the EMEM numbers to a scene snapshot configured in Kairos. Only the
snapshot names need to be provided.
```

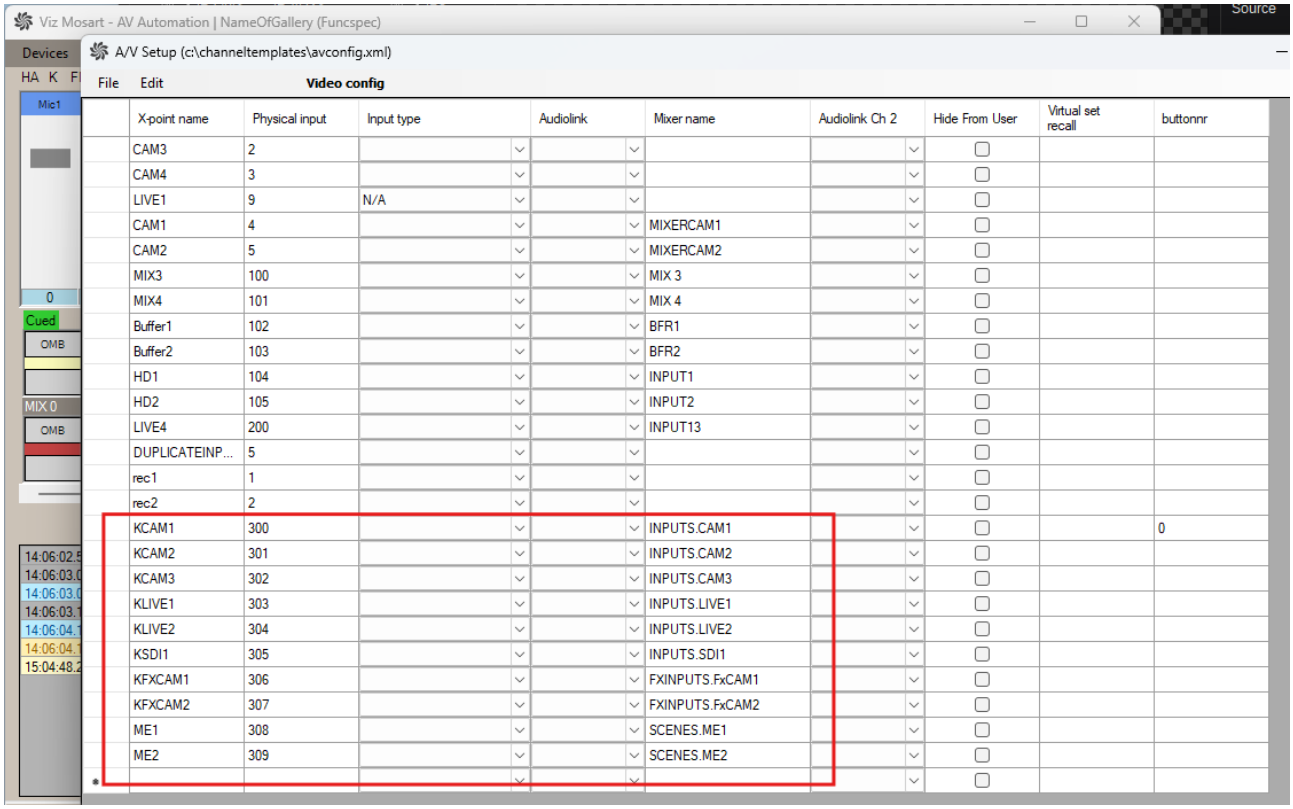
```
        As an alternative to these settings, you can rename the snapshots in Kairos
using numbers.
        When the EMEMs are triggered from a Mosart template, the full path is
automatically constructed by Mosart based on the Mixer Effect selected in the
template.

        When the EMEMs are used as effects (configured in AvAutomation > Devices > AV
Setup > Effects, columns EMEM Nr Preview/Program/Cleanup),
        the EffectMemory setting from the AvAutomation general settings is used to
build the full path.

        If an EMEM is to be recalled on several regions (scenes), then the same
snapshot must exist in all scenes this EMEM is to be recalled on.
        The setting name must follow the format "Emem" followed by a number,
corresponding to the EMEM number referenced in Mosart templates.
        -->
    <item name="Emem1" value="SNP1" />
    <item name="Emem2" value="SNP2" />
    <item name="Emem3" value="SNP3" />

    <!-- The path separator used in the commands sent to Kairos. If this is not a
dot, any dot found in the command will be escaped.
    When the dot is used as the separator, elements in Kairos (such as scenes,
layers, and input sources) must not contain dots in their names.
    The separator configured here should also be used when creating mappings in A/V
Setup or in this configuration file.
    For example:
        If the separator configured here is a slash (/), an input source named
"My.Cam1" in Kairos can be configured in A/V Setup as INPUTS/My.Cam1.
        If the separator configured here is a dot, the same input source must be
mapped in A/V Setup as INPUTS.My&#46;Cam1 (the dot is escaped using HTML encoding).
    Default: "." -->
    <item name="CommandsPathSeparator" value="." />

</Properties>
</DeviceConfig>
```

Crosspoints

To define Kairos crosspoints

- Open **AV Automation > Devices > AV Setup > Edit > Video config** tab and set up the crosspoints mapped to Kairos input sources following the example below (using the user-friendly names provided in Kairos).

> **Note**: The *full* path must be given with separator. The default separator is *dot*, which can be changed in KairosVideoSwitcherConfig.xml.



Effects and Mosart Templates

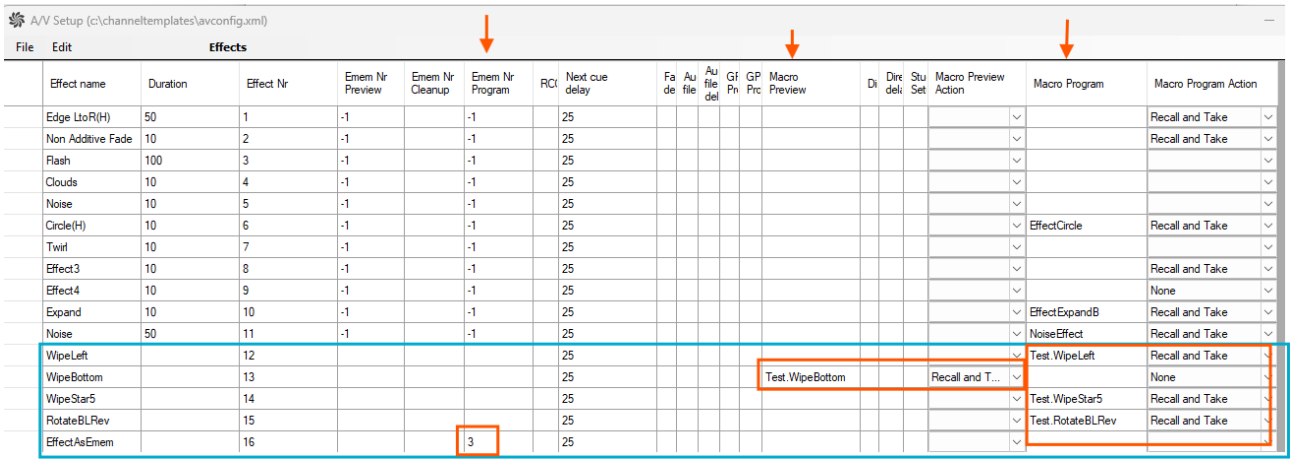
Effects

Vision mixer effects are managed in Viz Mosart as *macros*, using either the *global macros* or *scene macros* that have been created in Kairos.

To control a Kairos effect

1. Open **AV Automation > Devices > AV Setup > Edit > Effects** tab.
2. Add the full macro path.

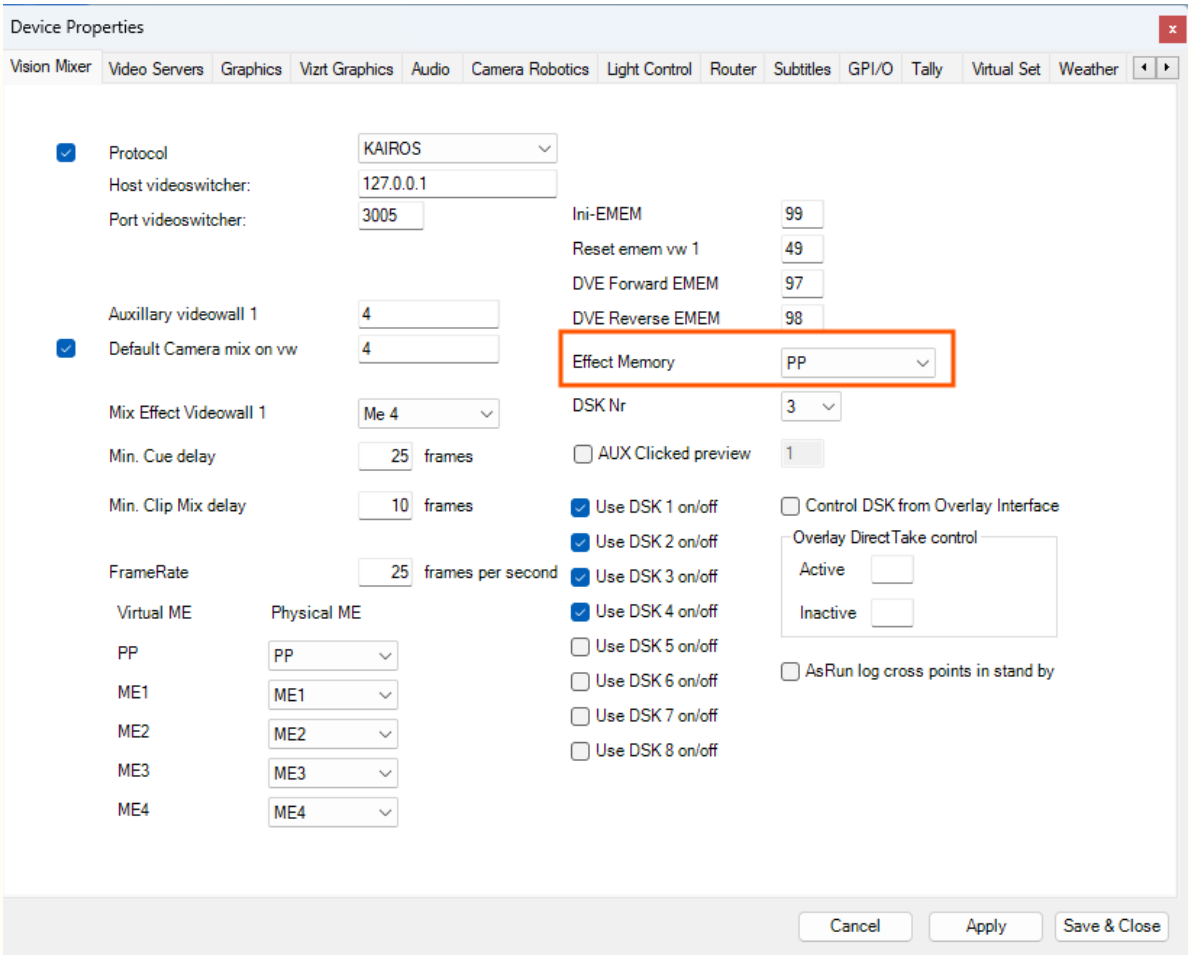
Example
In the screenshot below, the effects *WipeLeft*, *WipeStar5* and *RotateBLEmem* are configured as global macros to be executed on *Program*.
The effect *WipeBottom* is configured as global macro to be executed on *Preview*.



Effect name	Duration	Effect Nr	Emem Nr Preview	Emem Nr Cleanup	Emem Nr Program	RC(	Next cue delay	Fa de	Au file	Au file del	Gf Pn	GP Prc	Macro Preview	Di	Dire deli	Stu Set	Macro Preview Action	Macro Program	Macro Program Action
Edge LtoR(H)	50	1	-1		-1		25												Recall and Take
Non Additive Fade	10	2	-1		-1		25												Recall and Take
Flash	100	3	-1		-1		25												
Clouds	10	4	-1		-1		25												
Noise	10	5	-1		-1		25												
Circle(H)	10	6	-1		-1		25											EffectCircle	Recall and Take
Twirl	10	7	-1		-1		25												
Effect3	10	8	-1		-1		25												Recall and Take
Effect4	10	9	-1		-1		25												None
Expand	10	10	-1		-1		25											EffectExpandB	Recall and Take
Noise	50	11	-1		-1		25											NoiseEffect	Recall and Take
WipeLeft		12					25											Test.WipeLeft	Recall and Take
WipeBottom		13					25						Test.WipeBottom				Recall and T...		None
WipeStar5		14					25											Test.WipeStar5	Recall and Take
RotateBLRev		15					25											Test.RotateBLRev	Recall and Take
EffectAsEmem		16			3		25												

The effect *EffectAsEmem* was configured to recall a snapshot in Program, but to actually execute a *transition* you still need to provide a macro on Program.

> **Note**: EMEMs (snapshots) can also be recalled from an effect but *not* to play a transition.
> The **Effect Memory** setting from the AV Automation **Devices > Device Properties** (**Vision Mixer** tab) is used to build the full path for the snapshots recalled in this way:



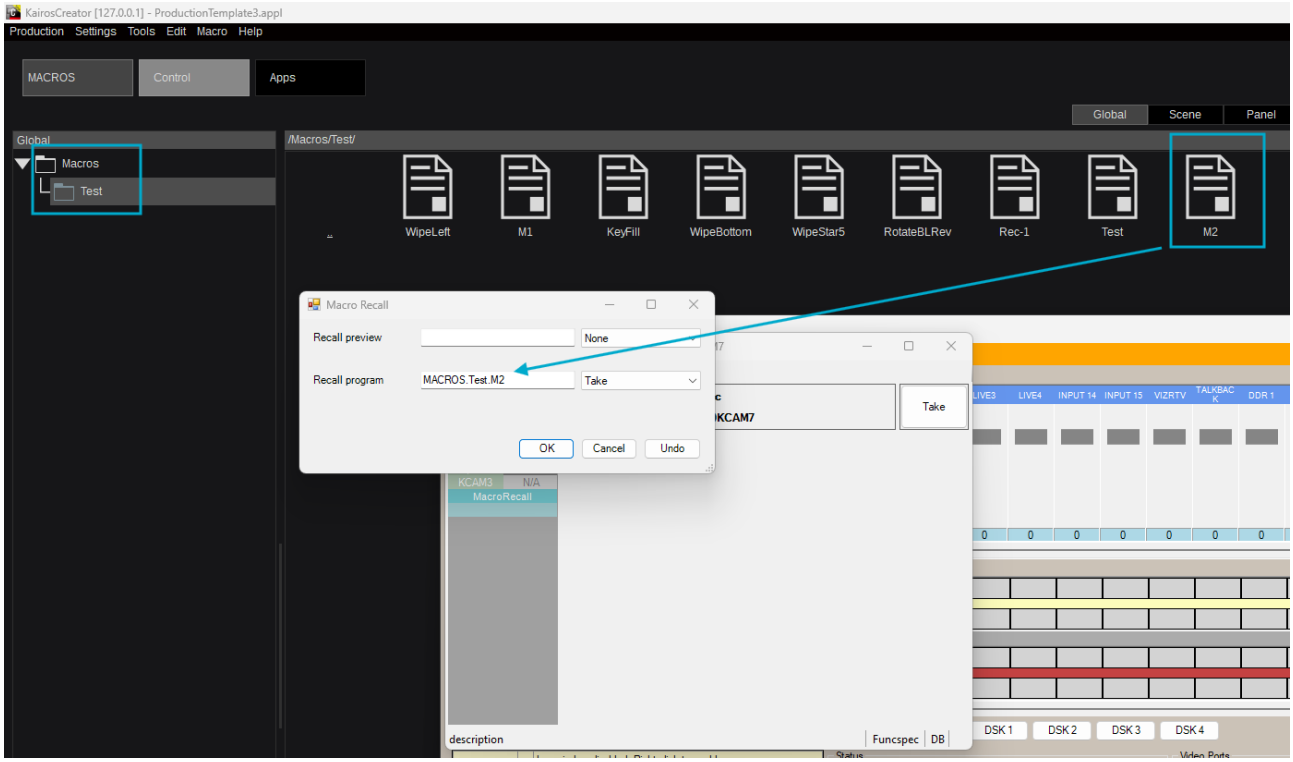
The EMEM number *3* is mapped in the Kairos config file to a snapshot created in Kairos.

Note: The snapshots are created per *scene*. For different regions, different snapshots need to be created.

Managing Macros

To recall a macro from a Mosart template

- Specify the full path to the macro you want to recall.



Note: When recalling a Mosart template from Kairos, there is no need to configure anything beforehand in the KairosVideoSwitcherConfig.xml (the macro device in the template can handle names as well as numbers).

12.1.7 Stagetec Driver Configuration

This section describes how to configure a Stagetec driver.

- AudioMixerStagetecDiamondRas.xml Config File
  - To change the Strategic config file
- Scenario Configuration
  - Sample config file Studio3Mics.xml
  - Understanding the Strip element
  - When FaderType=DeviceType
- Calling Scenarios with control commands

Setting up the Stagetec Driver

The configuration consists of three parts:

- If needed, edit the AudioMixerStagetecDiamondRas.xml config file
- Configure one or more scenarios
- Configure Mosart to use these scenarios.

AudioMixerStagetecDiamondRas.xml Config File

The default contents of this file are:

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConfig name="StagetecDiamondRasConfig">
  <Properties>
            <!--The strip number of the leftmost fader in the add-on panel-->
            <!--The default setting is for controlling strips 33-40-->
            <item name="FirstAddOnStrip" value="33"/>

            <!--This alternative setting is for controlling strips 1-8 of the Auratus
simulator-->
            <!--<item name="FirstAddOnStrip" value="1"/>-->

            <!--The initial scenario, if any-->
            <item name="InitialScenario" value=""/>
  </Properties>
</DeviceConfig>
```

- **FirstAddOnStrip**: Controls the add-on panel. Default *33*. Only change this value if necessary.
  The *alternative setting* is for controlling a Stagetec simulator.
- **InitialScenario**: Scenario that is used at start-up. Shall be a scenario name as described in section *Scenario Configuration* below.

To change the Strategic config file

Please follow the standard procedure:

1. Navigate to the installation folder (usually) at C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles
2. From ConfigurationFiles copy the Strategic xml to C:\channeltemplates.
3. Make changes to the *copy*.

Scenario Configuration

- One or more scenarios may be configured.
- Each scenario is represented by an xml file (with file extension .xml and containing valid xml).
- The scenario config files must be put in one of the places normally used for config files, preferably C:\channeltemplates.

Sample config file Studio3Mics.xml

The installation contains a sample scenario config file, *Studio3Mics.xml*. This file may be copied and used as a model for real scenarios:

```
<?xml version="1.0" encoding="utf-8" ?>
<StagetecDiamondRasStripsConfig>
  <Strips>
               <Strip Number="1" FaderType="DeviceType" Channel="MIC"
 PreviewOrProgram="Program" />
               <Strip Number="2" FaderType="Empty" PreviewOrProgram="Program" />
               <Strip Number="3" FaderType="Ignore" PreviewOrProgram="Program" />
               <Strip Number="4" FaderType="Static" Channel="Ch   3"
 PreviewOrProgram="Program" />
               <Strip Number="5" FaderType="DeviceType" Channel="MIC"
 PreviewOrProgram="Preview" />
               <Strip Number="6" FaderType="Empty" PreviewOrProgram="Preview" />
               <Strip Number="7" FaderType="Ignore" PreviewOrProgram="Preview" />
               <Strip Number="8" FaderType="Static" Channel="Ch   1"
 PreviewOrProgram="Preview" />
  </Strips>
</StagetecDiamondRasStripsConfig>
```

The outer parts are fixed:

```
<?xml version="1.0" encoding="utf-8" ?>
<StagetecDiamondRasStripsConfig>
  <Strips>
               ...
  </Strips>
</StagetecDiamondRasStripsConfig>
```

Understanding the Strip element

There are one or more *Strip* elements, each defining which channel to assign to the strip. The attributes are as follows:

Attribute	Valid values	Explanation
Number	1-8	Determines which strip is defined. Strips must be Numbered *1-8*. The actual strip number also depends on the value of FirstAddOnStrip described above, to give: Number - 1 + FirstAddOnStrip With the default config, Number=*1* then corresponds to strip *33*.
FaderType	*DeviceType* or *Empty* or *Ignore* or *Static*.	DeviceType: The actual channel will be determined as described below this table. Empty: The fader/strip will be unassigned. Ignore: The Strip element will be ignored altogether. Static: The channel will simply be the one with name equal to the value of the Channel attribute.
Channel	*Mixer channel name* or *device type* (none, server, server_ch2, graphics, external, mic, tlf, or soundplayer)	If (above) FaderType=*Static*: The mixer channel with the specified name is assigned to the stripe. If FaderType=*DeviceType*: The actual channel will be determined as described below.
PreviewOrProgram	*Preview* or *Program*	Used only if FaderType=*DeviceType* (but must be present also in the other cases, but then without any effect).

Sticky	*true* *false* (default)	Only required if FaderType=*DeviceType* and the fader allocation is reevaluated *after* the templates are changed. **Note**: This attribute is not considered when setting the *initial* scenario or when changing the current scenario with a *control command*. The effect of FaderType=DeviceType is described below and includes the quality of this *Sticky* attribute.

When FaderType=*DeviceType*

The actual channel is determined as follows:

- A channel of the type specified by the Channel attribute (an 'eligible' channel) will be assigned.
- If PreviewOrProgram=*Preview*, the channel chosen will be among the eligible channels in the preview template.
  If the scenario has *several* Strip elements with
  - FaderType=*DeviceType*,
  - the same Channel, and
  - PreviewOrProgram=*Preview*

each such Strip element will define a different channel (as long as there are enough eligible channels in the Preview template).

- Similarly, if PreviewOrProgram=*Program*, the channel chosen will be among the eligible channels in the program template. If the scenario has *several* Strip elements with
  - FaderType=*DeviceType*,
  - the same Channel and
  - PreviewOrProgram=*Program*

each such Strip element will define a different channel (as long as there are enough eligible channels in the Program template).

- If there are not enough eligible channels (in the Preview or Program template, respectively) of the specified type, each surplus strip will be *unassigned*, and have *no* fader/channel assigned.
- If the strip has the attribute *Sticky*, and the fader allocation *is* reevaluated *after* the templates have changed, the strip is left *as is* (this is the quality of *Sticky*).
- The Strip elements with FaderType=*Static* is not considered here. The same channel may be assigned to several strips.

Assuming that the Preview template contains the MIC channel *Mic1* (but not *Mic2* or any other) and that the Program template contains the MIC channel *Mic2* (but not *Mic1* or any other), the sample scenario results in this assignment:

Strip	Channel
33	Mic2
34	(unassigned)
35	(unchanged)
36	Ch 3
37	Mic1
38	(unassigned)
39	(unchanged)
40	Ch 1

Note: The fader allocation described above is reevaluated whenever the templates are changed.

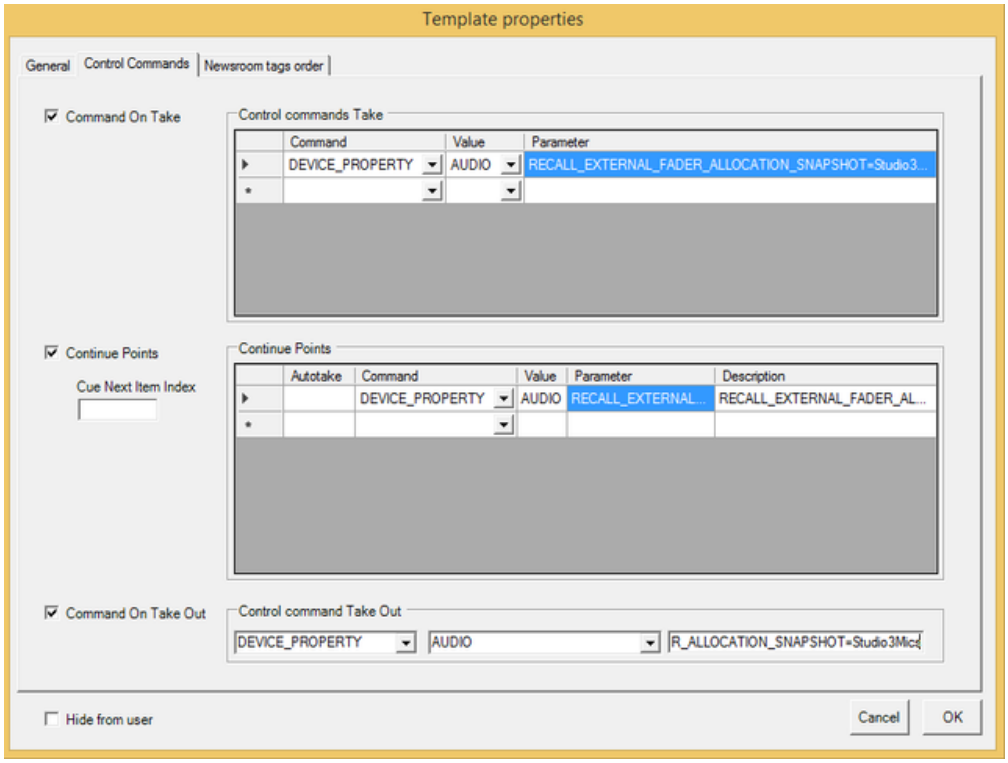
Calling Scenarios with control commands

Scenarios are invoked through *control commands*. These control commands should be structured as follows:

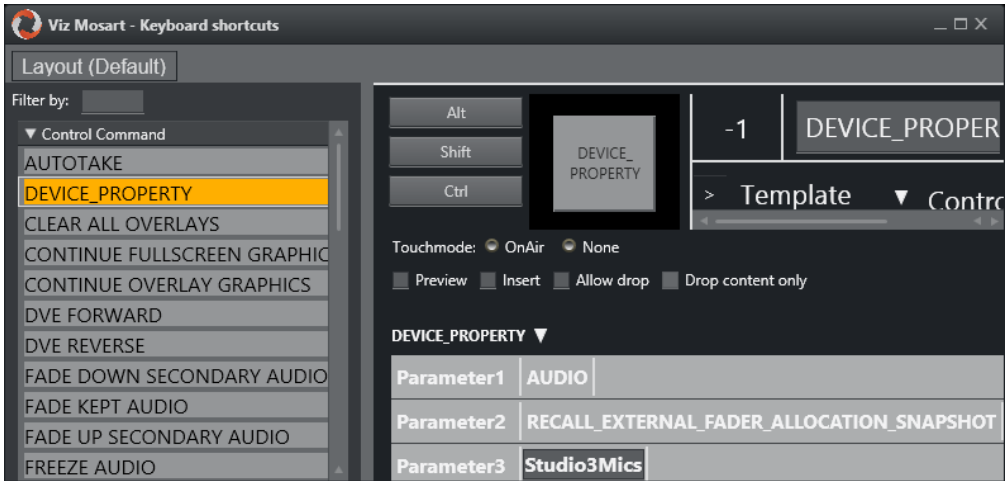
- **Command**: DEVICE_PROPERTY (usually selectable in a dropdown)
- **Value**: AUDIO (sometimes, but not always, selectable in a dropdown)
- **Parameter**: RECALL_EXTERNAL_FADER_ALLOCATION_SNAPSHOT=Studio3Mics.

Note: *Studio3Mics* is an example only. In practice, provide the current scenario name (file name *without* extension *xml*).

In **Template properties** it looks like this:


Template properties
General
Control Commands
Newsroom tags order
Command On Take
Control commands Take
Command
Value
Parameter
DEVICE_PROPERTY
AUDIO
RECALL_EXTERNAL_FADER_ALLOCATION_SNAPSHOT=Studio3...
Continue Points
Cue Next Item Index
Continue Points
Autotake
Command
Value
Parameter
Description
DEVICE_PROPERTY
AUDIO
RECALL_EXTERNAL...
RECALL_EXTERNAL_FADER_AL...
Command On Take Out
Control command Take Out
DEVICE_PROPERTY
AUDIO
R_ALLOCATION_SNAPSHOT=Studio3Mics
Hide from user
Cancel
OK

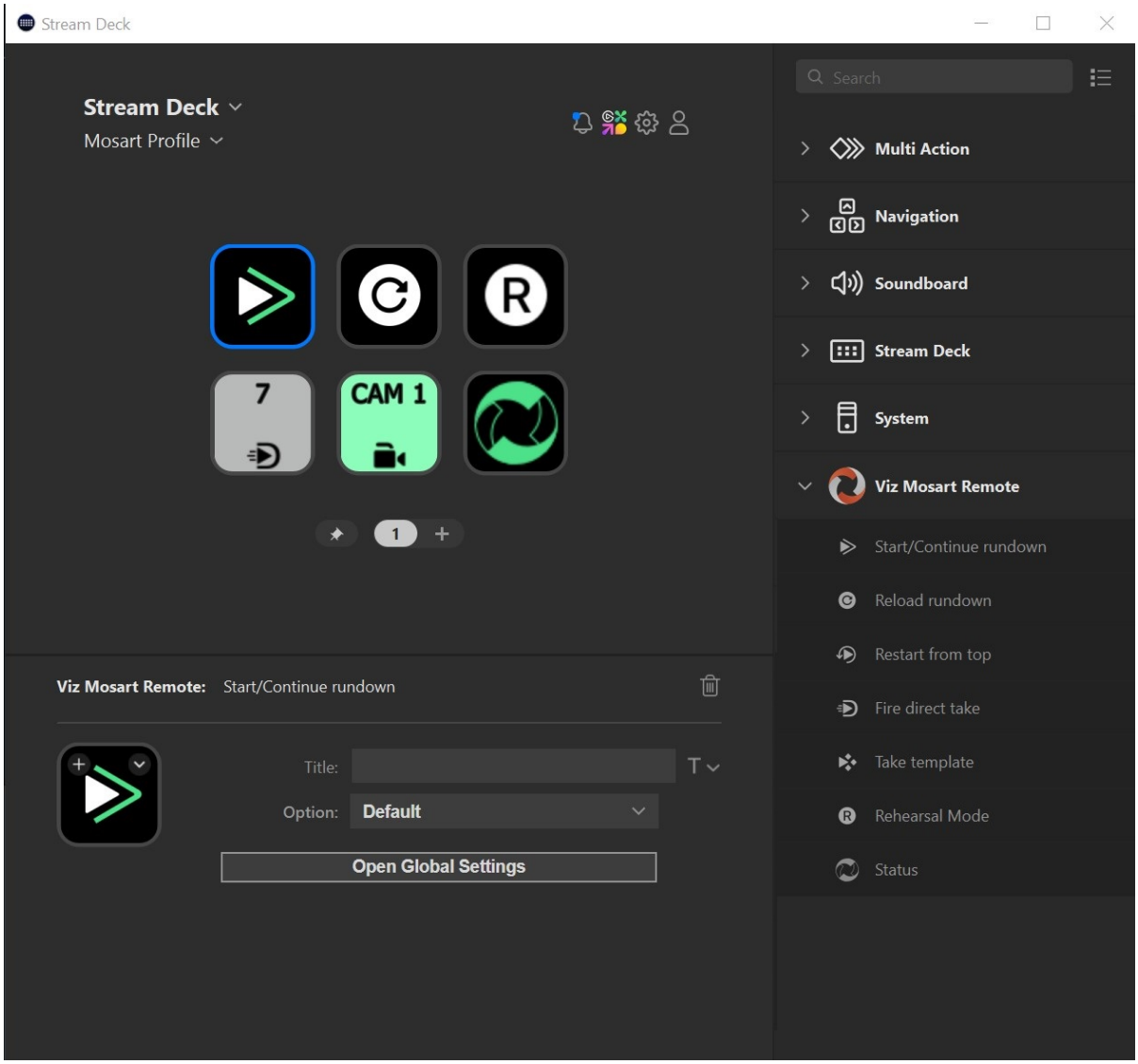
On the UI:


Viz Mosart - Keyboard shortcuts
Layout (Default)
Filter by:
Control Command
AUTOTAKE
DEVICE_PROPERTY
CLEAR ALL OVERLAYS
CONTINUE FULLSCREEN GRAPHIC
CONTINUE OVERLAY GRAPHICS
DVE FORWARD
DVE REVERSE
FADE DOWN SECONDARY AUDIO
FADE KEPT AUDIO
FADE UP SECONDARY AUDIO
FREEZE AUDIO
Alt
Shift
Ctrl
DEVICE_ PROPERTY
-1
DEVICE_PROPER
Template
Contro
Touchmode:
OnAir
None
Preview
Insert
Allow drop
Drop content only
DEVICE_PROPERTY
Parameter1
AUDIO
Parameter2
RECALL_EXTERNAL_FADER_ALLOCATION_SNAPSHOT
Parameter3
Studio3Mics

12.1.8 Stream Deck

The Elgato Stream Deck plugin for Viz Mosart allows you to trigger various Viz Mosart actions, such as Start/Continue the rundown, Take a template or firing a Direct take.

As a plugin to the Mosart Web Apps, this utility communicates directly with the Mosart server. All actions are triggered without needing a Mosart GUI open, active or in focus.



- For installation and operation instructions, please refer to the *Stream Deck* section of the Mosart Web Apps Guide.

12.1.9 TriCaster and Viz Mosart

This section gathers topics featuring Viz Mosart together with a Vizrt TriCaster.

- Connection Features
- Manual Configuration after Quick Connect
- File Overrides and Backups

Connection Features

- Viz Mosart connects to TriCaster for controlling multiple functions that include vision mixer, video server, audio mixer and sound player.
- Using TriCaster's REST API, Viz Mosart imports configuration settings into AV Automation video server settings or A/V Setup, enabling seamless automation.

This section explains how to quickly connect Viz Mosart to a TriCaster for immediate use.

For detailed configuration of video servers, switchers, audio mixers, and other devices, refer to the specific chapters in the documentation:

- Configuring a Vision Mixer (Video Switcher)
- Configuring a Video Server
- Configuring an Audio Mixer
- Configuring a Sound Player

If you are setting up a TriCaster *for the first time*, we recommend reviewing the topics above. However, if you need a quick way to establish a connection, follow the steps below.

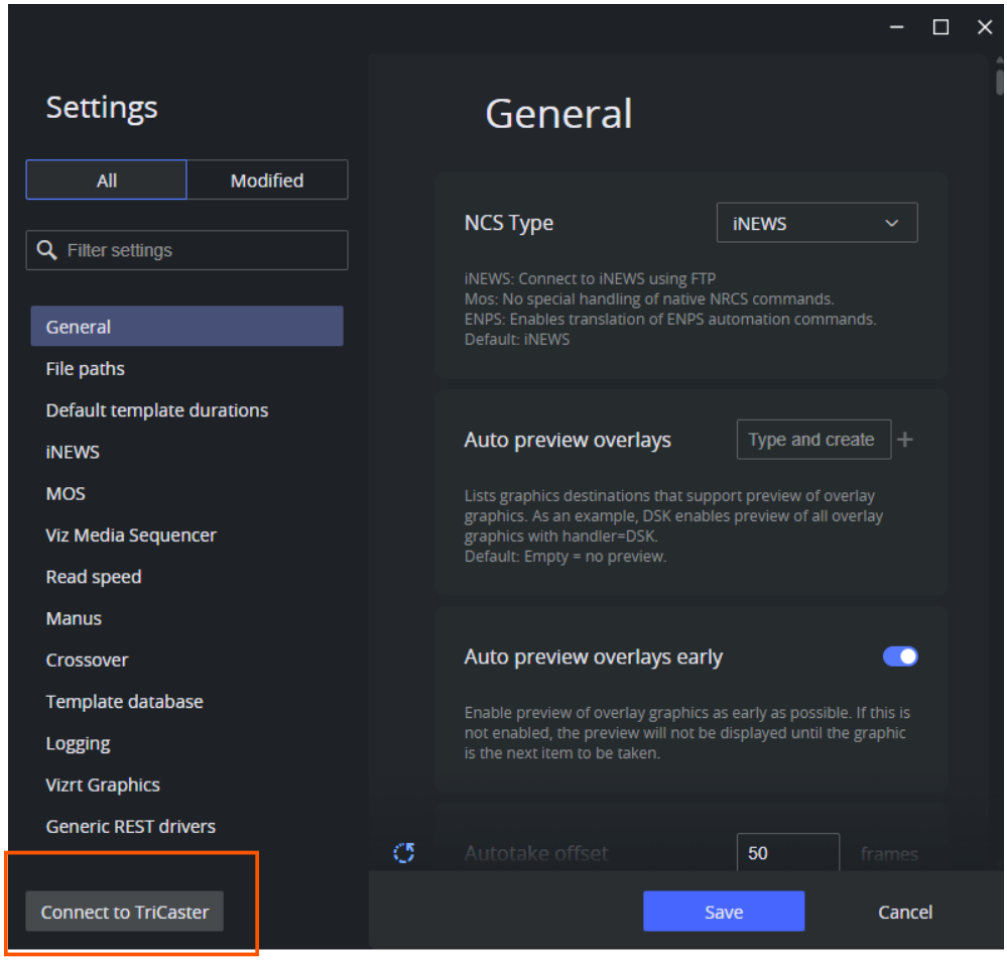
> **Info**: While the quick connection method imports TriCaster settings directly into Viz Mosart, small adjustments *may still be required* depending on how TriCaster was originally configured.
> After connecting, to ensure optimal performance, verify and fine-tune the settings in AV Automation from the menu bar **Devices > A/V Setup** and **Devices > Properties**.

To connect TriCaster to Viz Mosart

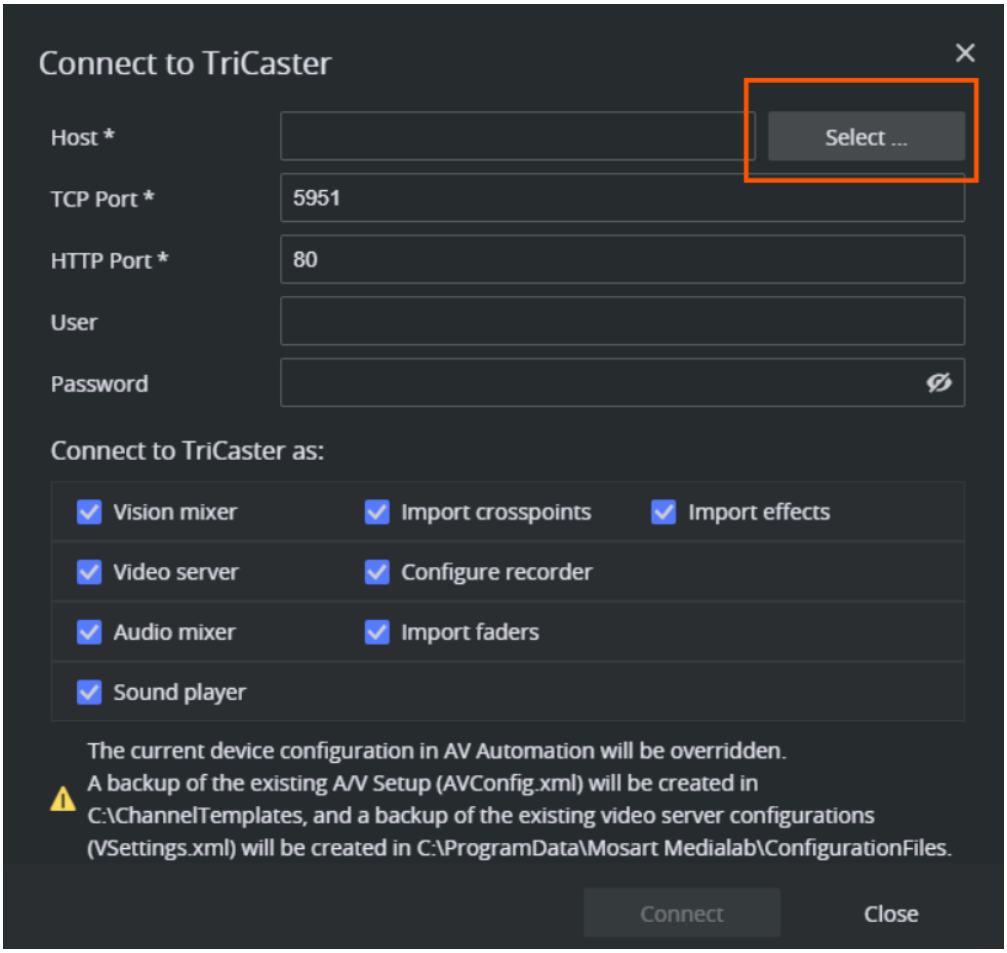
You use the common Settings menu, entering basic details about your TriCaster and its purpose in the system.

> **Info**: To connect, the TriCaster must be running an active session. An information message will advise on any issues encountered when attempting connection.

1. Open the **Settings** menu by either typing *settings* in the Manus Administrator console or selecting **Devices > Settings** from the AV Automation menu bar.



2. In the **Connect to TriCaster** menu specify:
   a. **Host**: Enter the TriCaster host name or IP address manually or browse NDI sources by using the **Select** button.
   Since all NDI sources are displayed without filtering, ensure you select the correct TriCaster source.



b. **TCP Port**: Used for control communication. Default: *5951*.
c. **HTTP Port**: Required for REST API access. Default: *80.*
d. **User** / **Password**: Enter TriCaster API credentials if required.

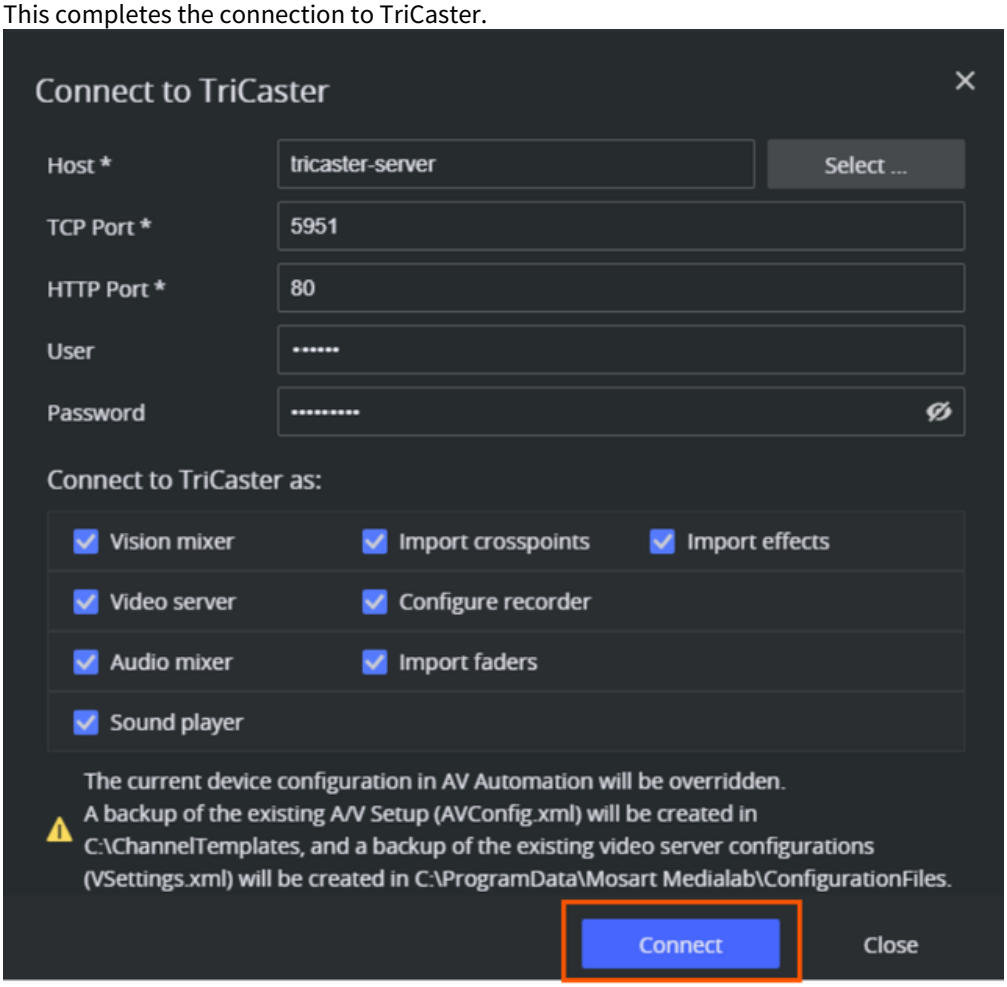
> **Info**: For password and security topics, please refer to the section Security.

4. In the panel **Connect to TriCaster as**, select the TriCaster features to integrate:
   a. **Vision mixer**: Enables Viz Mosart to control video switching on the TriCaster, with the option of importing crosspoints and effects.
   b. **Video server**: Enables control over video playback and recording on the TriCaster, with the option of configuring the recorder port.
   c. **Audio mixer**: Enables Viz Mosart to control TriCaster's audio, with the option of importing fader settings.
   d. **Sound player**: Enables control over sound playback on the TriCaster.

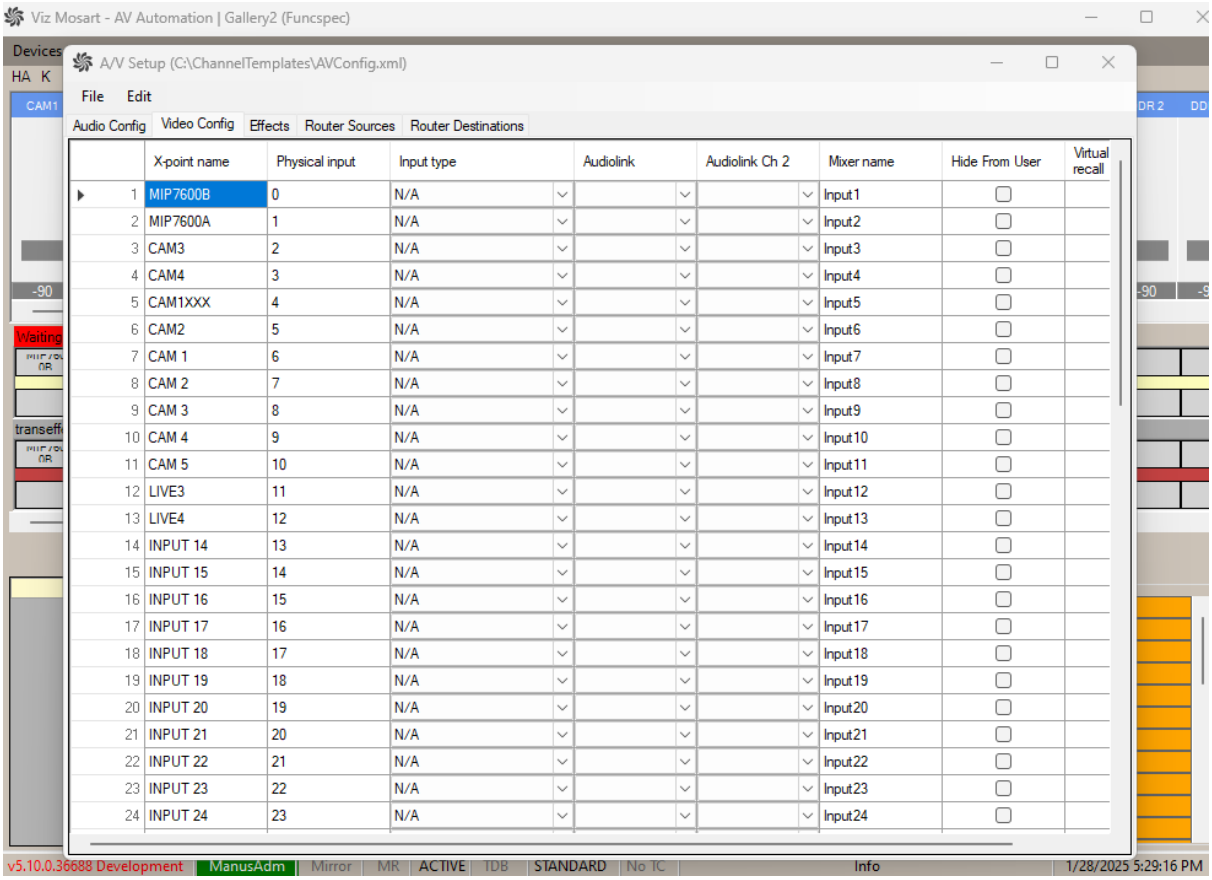
> **Info**:
> - For **Vision Mixer** and **Audio Mixer**, the imported configuration will *overwrite* the existing **Devices > A/V setup** (*avConfig.xml* file).
> A backup of the current *avConfig.xml* file is automatically created in the same Mosart configuration folder as where the active file is located.

- For **Video Server**, the changes will be applied in AV Automation under **Devices > Properties > Video Servers** tab (*vSettings.xml*).
  If a value for the TriCaster video server does *not* exist, it is added.
  If a value already exists, its settings are overwritten.
  A backup of file *vSettings.xml* is created in the Mosart configuration folder where the active file is located.

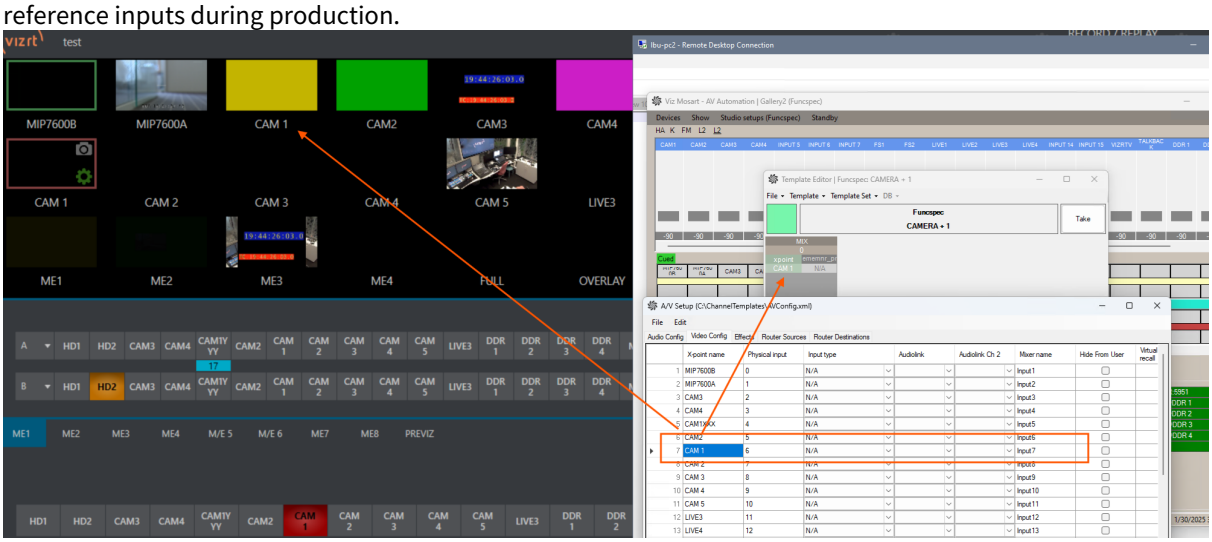
5. Once **Host**, **TCP Port**, and **HTTP Port** have valid entries, click **Connect**.
   This completes the connection to TriCaster.



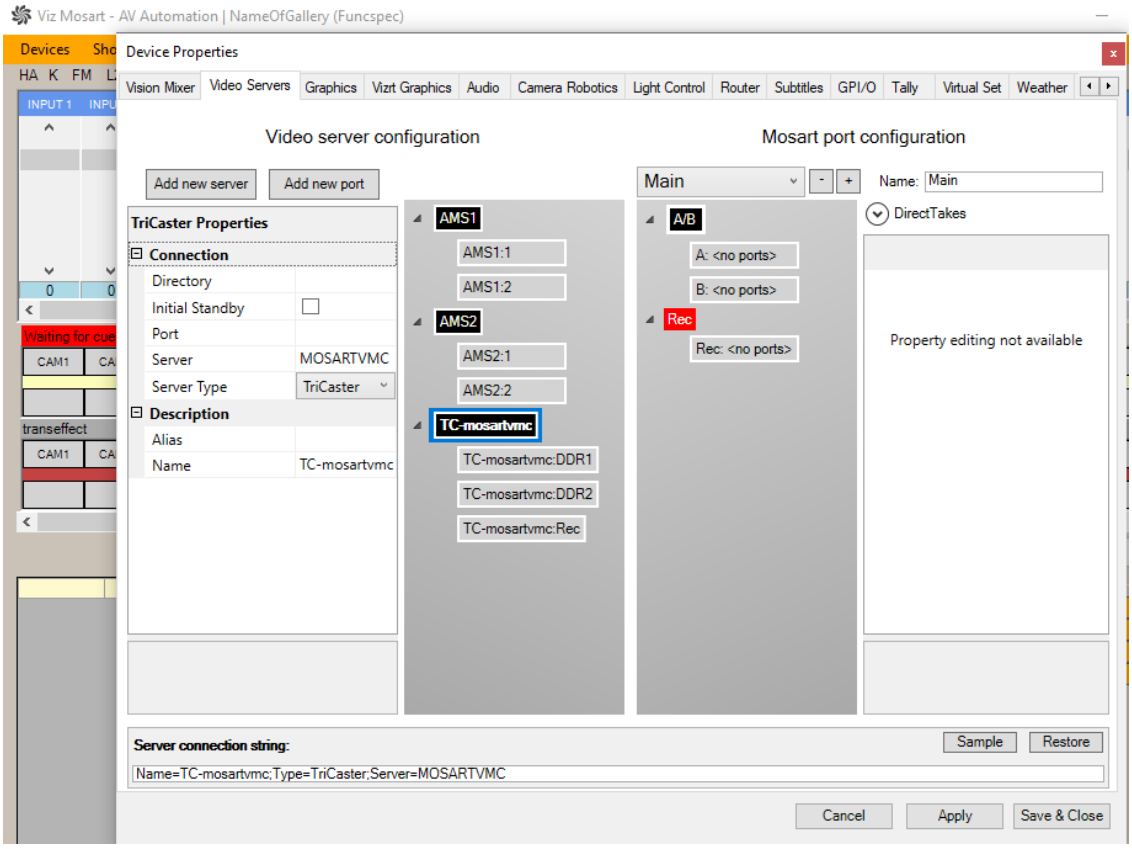
6. Review the **A/V Setup** settings in AV Automation under **Devices > A/V Setup**, the Audio Config and Video Config tabs.



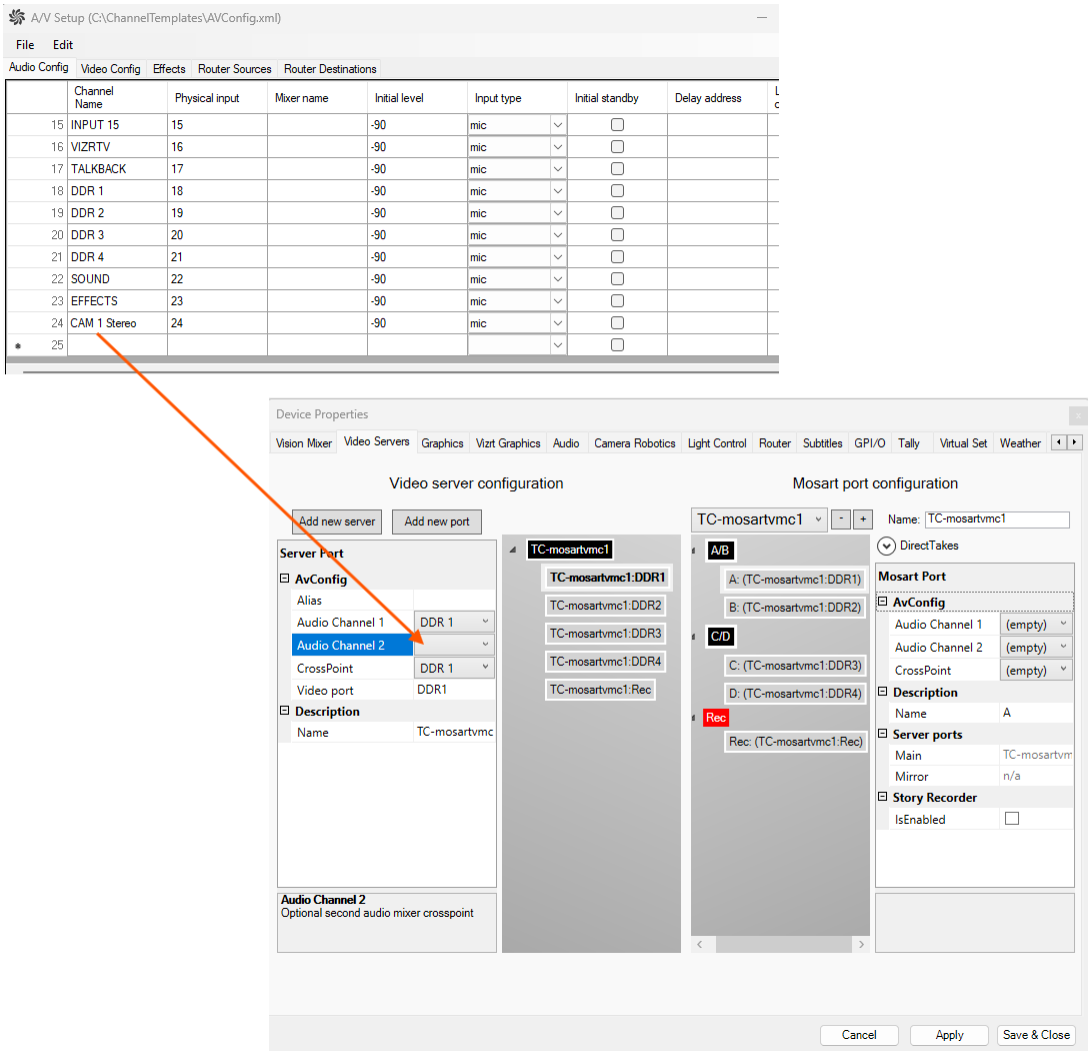
To minimize further changes in Viz Mosart after importing crosspoints and audio channels, it is recommended to configure TriCaster with *user-friendly names* for video and audio inputs. These names are displayed in Viz Mosart, including the Template Editor and the Mosart UI, making it easier to manage and reference inputs during production.



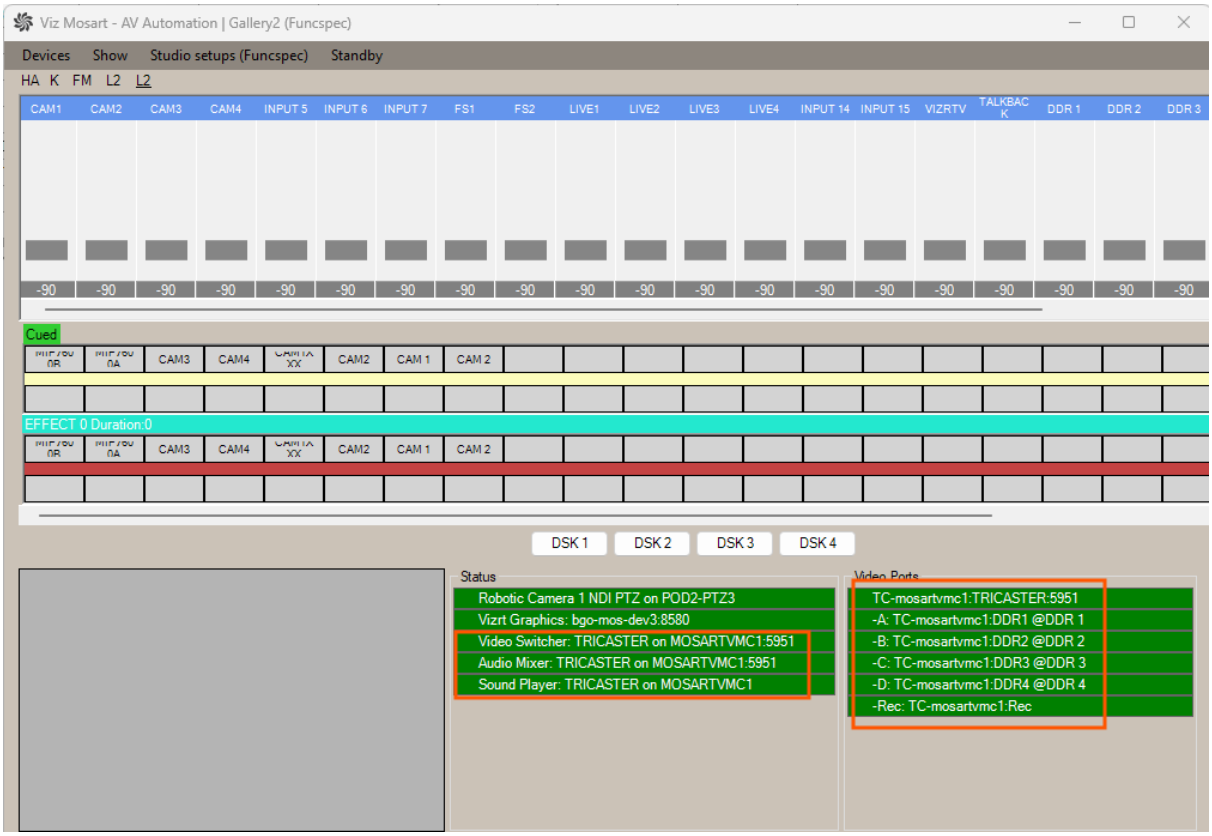
7. In AV Automation, under **Devices > Properties > Video Servers**, review the video server settings.
   a. When **Video server** is selected: A TriCaster video server named *TC-{TriCasterHost}* is automatically added (see screenshot below) or, if a configuration for that host already exists, it is overwritten.
   b. When **Configure recorder** is selected: A virtual port named *Rec* is generated. This is controlling the global recorder enabled for



If **Audio Channel 2** (the second mono channel in a stereo pair) is used, an audio channel crosspoint must be manually configured in **A/V Setup** that matches the corresponding audio input in TriCaster:



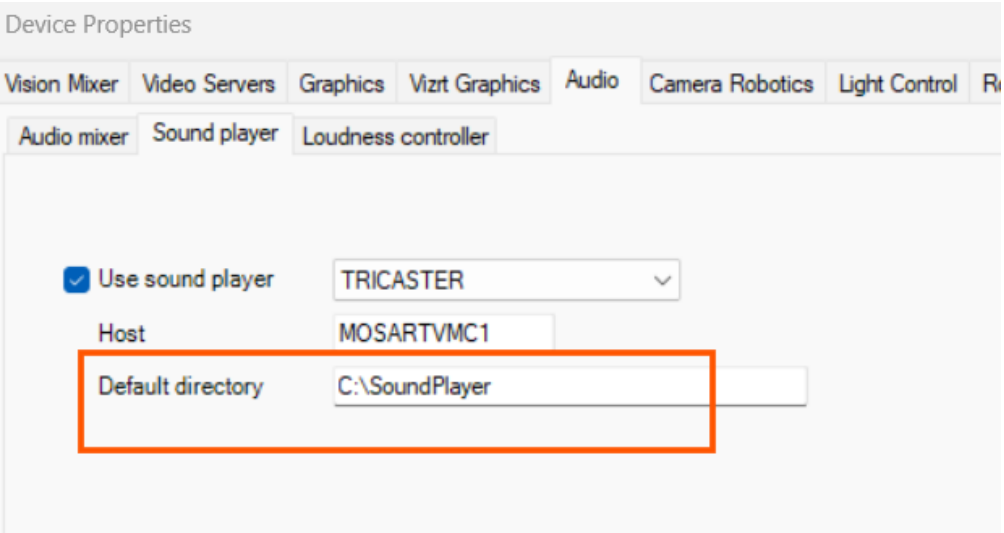
4. If the *audio* or *video* channel *crosspoints* were imported, restart AV Automation for the new configurations to take effect.
5. Verify connection status in the AV Automation UI:



Manual Configuration after Quick Connect

Quick Connect automates most of the TriCaster setup in Viz Mosart. However, some configurations must still be done manually:

1. **TriCaster session monitoring**
   a. Make sure you are in the correct TriCaster session when running a show from Viz Mosart connected to TriCaster.
   b. If unsure, re-run the TriCaster Quick Connect dialog to verify the session.
   c. The session name is only displayed in the Quick Connect feedback message and not stored anywhere in Viz Mosart.
2. **EMEM path configuration**
   a. The path to EMEMs is not automatically set and defaults to *D:\Sessions\SessionName\emem* in the *TriCasterVideoSwitcherConfig.xml* file located at
   Mosart installation folder > Mosart Server > ConfigurationFiles.
   b. Copy this file into the *%ProgramData%\Mosart Medialab\ConfigurationFiles* and set the path accordingly.
3. **Sound Player default path**
   - The path to the sound clips directory on the TriCaster machine must be manually adjusted after connection:



4. **Mosart Media Administrator setup for clip monitoring**
    - Clip monitoring is not automatically configured and must be performed manually. For an example, see section Viz One and Media Service Connection String.
5. **User adjustments for crosspoints and video server ports**
    - If the TriCaster input sources for audio and video were configured with non-friendly names, users must manually adjust them in A/V Setup as described above.

These manual adjustments ensure full integration between Viz Mosart and TriCaster. Future updates may further automate the process.

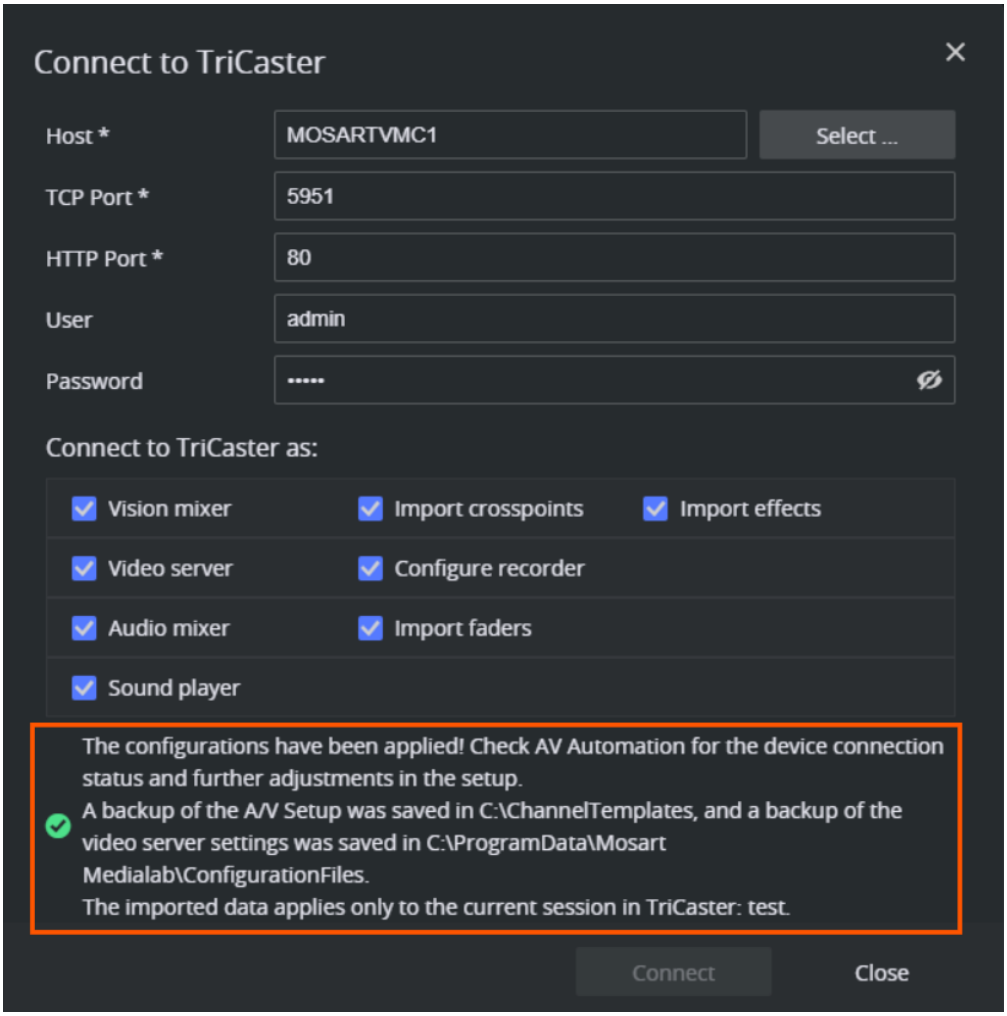
File Overrides and Backups

When successfully connected to TriCaster, any existing configurations in AV Automation and relevant files (such as *AVConfig.xml* and *VSettings.xml*) are overridden with your input from this **Connect to TriCaster** menu.

Automatic backup

At file override, to retain the previous data a backup file is generated in the same folder as the overridden file.

- Details are presented in the Information field at the foot of the menu.



This backup feature enables simple reconfiguration of the TriCaster after connecting, no matter whether the connection is successful or not.

Info: Information messages relate to the current, enabled TriCaster session only.

12.1.10 Adaptive Live Producer

- Overview
- Features
- Abbreviations and Key Concepts
- In this Section

Overview

Adaptive Live Producer (ALP) lets Viz Engine act as the *main content compositor for all production sources*, under the exclusive control of Viz Mosart. Rather than just routing signals like a traditional switcher, Viz Engine composes live sources, DVE effects (box effects, picture-in-picture, butterflies, etc.), fullscreen and overlay graphics into a single program output. With ALP, you build an *IP-based workflow* that uses the power of Viz Engine to *enhance the production with rich animations, layouts, and custom looks*. The full show output is built as a single Viz Engine graphics composition that handles live sources, graphics, layouts, and transitions.

ALP runs under the exclusive control of Viz Mosart, but is not an integrated single unit. Viz Engine and Viz Mosart run in their own environments, and Viz Mosart templates drive a broad scope of traditional and unique switcher features. On the Viz Engine side, ALP uses *superchannels* to hold live sources, graphics and clip playback. Each superchannel has two internal sub-channels (A/B), so sources can be swapped with animated transitions for added flexibility and performance.

ALP succeeds the current Viz Engine as a Switcher, and is the way forward for running Viz Engine as a switcher in Viz Mosart, now driven through the Media Sequencer (MSE). Operators keep working the way they already do. Stories are cued and taken from the rundown, existing templates keep working, and the layout and look on air come from *presets* defined in the scene.

> **Info:** Early preview. Adaptive Live Producer is an early preview feature in Viz Mosart 5.15. It depends on beta (pre-releases for selected customers) versions of Viz Engine, Viz Multplay and MSE that are not yet generally released, and it will gain more capabilities in later releases of Viz Mosart and the related applications. To evaluate ALP, contact Vizrt Support for the compatible component versions and setup guidance.

When to use ALP

ALP is built for *creative production*: shows where the program look is driven by graphics, animations, virtual sets, and custom transitions. Because Viz Engine itself is the switcher, the layout on air *is* a graphics scene, and the things a traditional hardware switcher cannot do become first-class:

- **Rich graphic animations** as part of presets and preset transitions, not only hard cuts and crossfades.
- **Shaders and custom looks** designed in Viz Artist (3D backgrounds, lighting, branded effects).
- **Custom content transitions** authored as directors, per superchannel and per direction.
- **Built on Viz Engine**, where virtual studio, AR and other 3D scene work are part of the same workflow.
- **Tight integration** with the other Vizrt solutions (Viz Mosart, Viz Pilot Edge, Media Sequencer).

ALP is not aimed at replacing a hardware vision mixer where the show is essentially ‘route the program through’. A traditional switcher with purpose-built frame-accuracy, high reliability and inexpensive hardware switching, remains the right tool for this.

- Use a hardware switcher for channel branding, playout, and master control workflows where the value is routing a finished signal.

Copyright © 2026 Vizrt

- Use ALP when the value is the creative composition itself.

How ALP differs from the existing Engine Switcher

The current Viz Engine as a Switcher holds its switching logic inside a custom script in the Viz Engine scene. This approach constrains graphic design and requires resources to support and upgrade.

ALP reaches the same goal with standard Viz Engine scenes built on superchannels, a powerful Viz Engine compositing feature that holds and switches all live sources, graphics, and clips, with no custom switching script in the scene. Everything runs under standard Viz Mosart control through the Media Sequencer. The result is more robust, easier to support, and easier to extend.

Features

Adaptive Live Producer in Viz Mosart 5.15 drives graphics and live sources through Viz Engine. All features below come with this version, any current exceptions are stated.

Video switching

- Preset switching (1Box, 2Box, 3Box, BigSmall, …).
- Live source switching on the background (SC1).
- Set sources on split-screen boxes (any superchannel).
- Synchronized preset and content switching (under the conditions in Templates and Show Design).
- Preset transitions (cut, or animate with duration).
- Content transitions on the background, Cut, Mix, Wipe, and Effects (on the main background, i.e. SC1)
- Multiple mixer effects (PP, ME1, ME2, …).
- Delay the mix and the transition on the main background (SC1).
  Works only with *Mix Delay,* and *Transition.*
- Keyboard shortcuts and control commands (INIT, CLEANUP, SET_PRESET).
- Clip playout.
  Works only with an external video server output entering as a live source into the Viz Engine, which in turn is acting as a switcher.
- Connection status: Only the MSE connection is presented, *not* the individual Viz Engine connections.

Fullscreen graphics

- Take a graphic on a specific superchannel.
- Take a graphic in sync with other content (under certain conditions).
- Cue and Preview; Continue (advance continue points).
- Static graphic from a template.
- Graphic content from the NRCS (as a Viz Pilot Edge MOS object).
- Take out graphics with or without delay from a superchannel.
- Prepare, Recue, accessory templates, initialize/cleanup playlist.
- Fullscreen graphics concepts (design variants).
  For this version, set with a macro.

Tally: supported over TSL UMD v3.1. See Tally in the Adaptive Live Producer for more information, and Tally on how to set it up.

Abbreviations and Key Concepts

Term	Meaning
ALP	Adaptive Live Producer, a Viz Mosart switcher driver that turns Viz Engine into a software production switcher, driven through the Media Sequencer.
Superchannel (SC)	An abstraction on top of a Viz Engine media channel. Superchannels support various asset types like images, graphics, clips, streams, live content. Each superchannel has two sub-channels, so content in each channel can be played out over an A/B workflow. One sub-channel (A) is active, while the second (B) is inactive. New content can be prepared on the inactive sub-channel, a transition effect can then run the transition from the active to inactive sub-channel, switching their roles.
PP	Program/Preset, the main mixer output that goes to program.
Mixer Effect (M/E)	A Mix Effects (M/E) bank is the row or bank of source-selection buses on a vision mixer that are mixed into a single output. One M/E groups a *Program* bus (what is on air), a *Preview* bus (what is cued next) and one or more *Key* buses (keyers layered on top); a larger switcher has several M/E banks.
Video wall (Viz Multiplay)	A Viz Multiplay configuration object. In Viz Multiplay a video wall normally describes a physical screen layout built from Viz Engine's superchannels. In an ALP setup its main role is to produce the program (PGM) output for broadcasting a show, the way a switcher does. It can still target a physical video wall, but that wall is filled through Viz Mosart templates and actions, not from Viz Multiplay. In ALP there is one video wall per Viz Engine output (one Mixer Effect: PP, ME1, ...). These pages use *Mixer Effect*, and mention a video wall only when describing the Viz Multiplay configuration.
KeyFill	A fill source placed on a split-screen box, on superchannels SC2 and up.
DVE	Digital Video Effect: a split-screen / box layout such as 2Box or 3Box. This is the broadcast meaning used across the ALP pages, and is not the same as the Viz Engine *DVE mode* below.
DVE mode (Viz Engine)	A Viz Engine output/render mode for the Matrox card, used in the clean and dirty setup and paired with *Texture mode*. It is unrelated to the broadcast DVE (box layout) above. See Clean and Dirty Output.
Texture mode (Viz Engine)	A Viz Engine render mode where the program is composited as a texture; used for the clean engine when animations, shaders or custom transitions are needed. Paired with DVE mode in the clean and dirty setup.
FGX	Fullscreen graphics.

Copyright © 2026 Vizrt

Term	Meaning
Preset	A named layout and look (for example 1Box, 2Box, 3Box, BigSmall), defined in the scene. Mosart fills presets with content and takes them to program.
Preset transition	Animates the change of layout (for example 1Box to 2Box).
Content transition	Animates the change of the source playing on a superchannel (for example a crossfade on the background). These can be custom transitions or transition shaders.
Concept	A design variant of a graphics scene, for example a branded look per show, channel, or region.
Director	Content transition director in Viz Engine drives animated transitions between content on a superchannel.
MSE	Media Sequencer. The engine through which Mosart controls Viz Engine. ALP plays out through the Media Sequencer using its superchannel arm/fire mode. For the full list of Media Sequencer commands and their behavior, open the profile command reference on your own MSE installation (replace the host with the address of your MSE server) at: *http://mse-server:8580/doc/profile_command.xhtml#command_for_superchannel_arm_fire*
VDOM	Virtual Document Object MOdel . This is the Media Sequencer's data tree, browsable from the MSE home page at *http://mse-server:8580/app/vdomconfig/vdomconfig.html*. More documentation can be found at *http://mse-server:8580*.
Arm	An MSE command which prepares a preset or a content element so it is ready to play out, without putting it on air yet.
Fire	An MSE command which plays out a single armed element on its channel.
Fire all	An MSE command which plays out all armed elements together in the same frame, so a preset and its content switch synchronously.
Take	An MSE command which puts an element straight to the output, without arming it first.
Cue	An MSE command which shows the first frame of an element.
Out	An MSE command which removes an element from the output.

Copyright © 2026 Vizrt

Term	Meaning
KBS	Keyboard shortcut.
NRCS	Newsroom Computer System, the rundown source.
SFP	Small Form-factor Pluggable. It is a compact, hot-pluggable network interface module format used for both telecommunication and data communications applications.
CLN	Clean output. See Clean and Dirty Output.
PGM	Program output. Also referred as dirty output. See Clean and Dirty Output.

In this Section

The following pages introduce ALP in detail:

- Setup and Configuration: End-to-end setup from Viz Engine scenes through Multiplay, the Settings app, AV Automation, and A/V Setup, plus the connection check.
- Templates and Show Design: Template configuration (Switcher Plugin, graphics on superchannels, graphics concepts), keyboard shortcuts, control commands, synchronization rules, and the end-to-end verification.
- Clean and Dirty Output: How to produce a clean feed (without overlay graphics) alongside the dirty program feed.
- Tally in the Adaptive Live Producer: Why ALP relies on Mosart-generated tally and the ALP-specific rules, with a reference to the Tally section for protocol and setup.
- Troubleshooting: What to collect when reporting an issue.

Copyright © 2026 Vizrt

Setup and Configuration

This is an end-to-end setup for Adaptive Live Producer (ALP). Follow the sections in order. The setup spans Viz Engine, Viz Multiplay, the Media Sequencer (MSE), and Viz Mosart.

- Key Concepts
- Prerequisites
- Setup Overview
  - Engine Configuration
- Viz Engine Scenes
  - Mixer Effect Setup
  - Preview Setup
- Viz Mosart Setup
  - Device setup for Adaptive Live Producer
  - A/V Setup for Adaptive Live Producer
  - Verify the Connection

Key Concepts

Viz Mosart drives Adaptive Live Producer using the same vision-mixer concepts it uses for a traditional hardware switcher. Knowing how those concepts map to ALP makes the rest of the setup easier to follow.

- **Mixer Effect**
  In a traditional switcher a Mixer Effect (M/E) is an independent mixing stage that produces one output. In ALP each Mixer Effect maps to a *Viz Engine instance*.
  PP (Program/Preset) is the main program output; *ME1, ME2,* **...** are additional outputs.
- **Main output and Keyers**
  Superchannels are the content layers inside a rendering engine. They are the equivalent of the *buses* and *keyers* of a traditional switcher:
  - *SC1* is the background (the program/background bus)
  - *SC2…SCn* are the keyer / key-fill layers drawn on top.
- **Layout**
  A *preset* is the on-screen layout of the superchannels (the DVE box layout). Selecting a preset is the equivalent of recalling an E-MEM / snapshot that arranges the boxes on a traditional switcher.

> **Info**: *Superchannel 1 is* the background by *role*, not necessarily by *look***.** Mosart treats SC1 as the background bus, and commands such as the *Switcher Crosspoint* always target SC1. The scene you design for a mixer effect can still carry its own main background, with every superchannel (SC1 included) acting as a keyer on top of it. In that case SC1 is not the visual background; it is simply the layer that background-targeting commands land on. Design the scene with this mapping in mind: whatever a *Switcher Crosspoint* drives goes to SC*1*.

Copyright © 2026 Vizrt

ALP concept	In Mosart	Traditional switcher equivalent
A Viz Engine with the *Preset Plugin* enabled (is using superchannels)	Mixer effects: PP, ME1, ME2.	A Mix Effects (M/E) bank, the row or bank of source-selection buses on a vision mixer that are mixed into a single output. One M/E groups a *program* bus (what is on air), a *preview* bus (what is cued next) and one or more *key* buses (keyers layered on top); a larger switcher has several M/E banks.
Superchannel SC1	Background content controlled using: • Switcher Crosspoint • KeyFill SC 1 • Graphics SC 1.	Program / background bus of a M/E. See also the Info box above.
Superchannels SC2…SCn	KeyFill SCn, n >= 1	Keyers / key-fill buses.
Superchannel visibility	Only with preset change using: • Switcher Plugin on a template; • SET_PRESET control command on a template or keyboard shortcut.	Keyer visibility.
Preset (1Box, 2Box, …)	DVE layout controlled via Switcher Plugin	E-MEM / snapshot of the DVE layout.
Content transition: none	Mix 0 on a Switcher Crosspoint. Cut option on a KeyFill.	Hard cut of the source, on the program (background) bus or on a keyer fill.
Content transition: custom or shader	Mix/Wipe rate > 0. Effects (background). No transition control on *KeyFill* commands for now (scene content transition applies).	An animated source change. On the background, a mix from the preview bus to the program bus of the M/E. On a keyer, an animated source change, as supported by some switchers.

Prerequisites

Component	Minimum version
Viz Engine	5.5.2
Viz Multiplay	0.0.0.44610 beta
Viz Mosart	5.15.0
MSE	MediaSequencer.BundleInstaller-0.0.0.33515-Beta

Note: All Viz Engines in the setup must run the same version and share the same presets, content transition directors, and superchannel count.

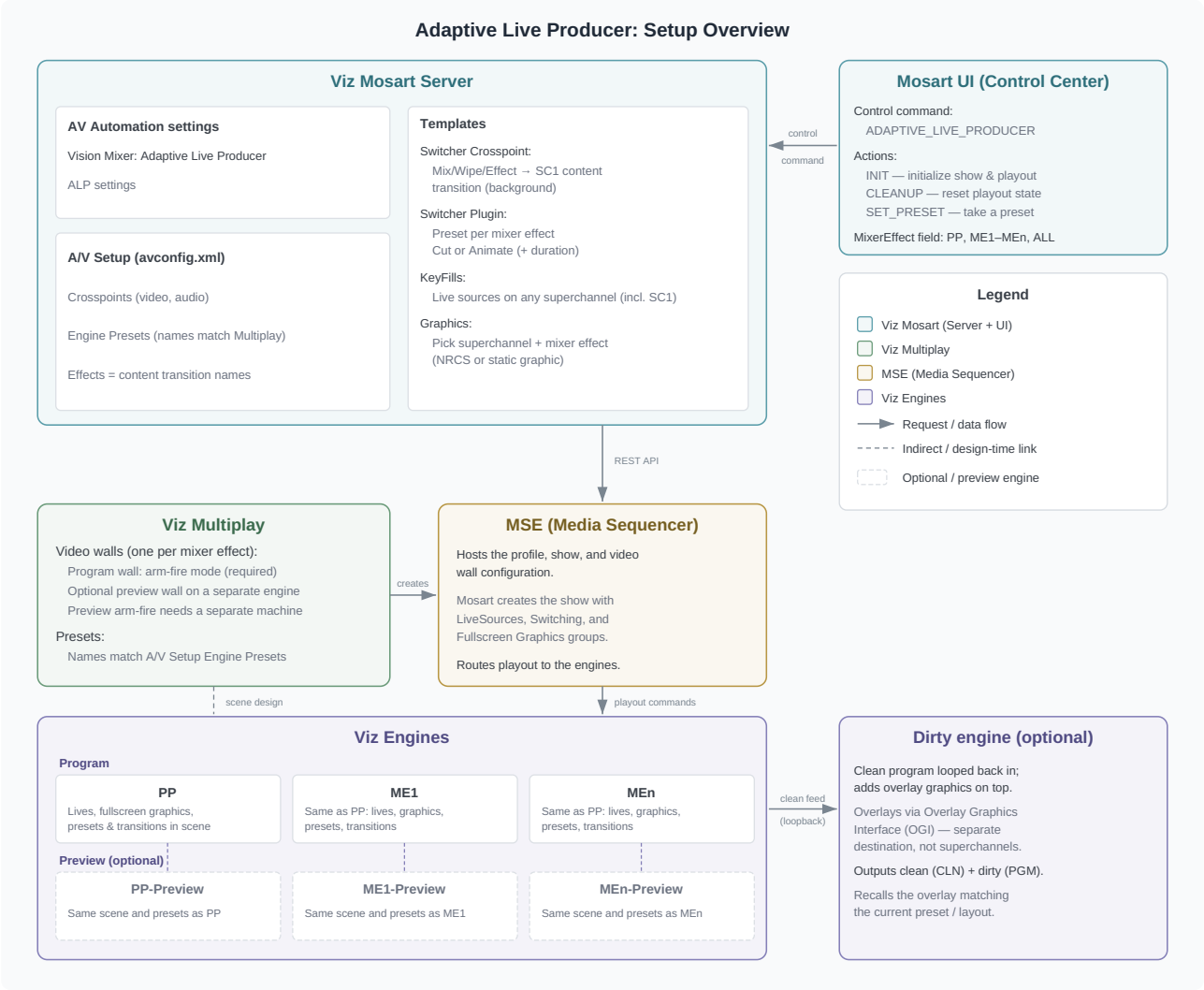
Setup Overview

Before you build the setup, plan it around your production needs:

- **How many mixer effects (outputs) do you need?** PP is the main program output; add ME1, ME2, … only when you need additional independent outputs. Each mixer effect is a separate Viz Engine instance, so more outputs means more engines.
- **Do you need preview, and for which outputs?** Preview is optional and decided per mixer effect. A preview output needs its own engine, and cue-time transitions only replicate on preview when that engine runs on a separate machine. See Preview setup below.
- **Do you need a clean and dirty output?** If you must deliver a clean feed (without overlay graphics) alongside the dirty program feed, plan a second (dirty) engine per output that adds the overlays. See Clean and Dirty Output.
- **Size the engines.** Decide the number of superchannels and live sources, and make sure every engine that serves a mixer effect, shares the same scene, presets and content transitions.

Info: *No separate M/E is needed to build a DVE box layout.* On a traditional switcher, a DVE / box layout is built on its own M/E and then projected into program (M/E-in-PP).

- In ALP, the MSE arm-fire workflow together with the A/B (program/preview) role of each superchannel, lets you compose and take a DVE layout *directly on PP*.
- Only add an additional mixer effect (*ME1, ME2*, …) when you purposefully need a separate independent output, and not when just building a DVE.



Engine Configuration

Scene cleanup

By default, Viz Engine automatically frees (cleans up) scene content held on a superchannel sub-channel once it is no longer active. In an ALP setup this auto-cleanup conflicts with the A/B background-loading workflow. When a new graphic scene, or a high-res/compressed image, is loaded on the inactive sub-channel (in preview/background) while program is on air, the engine can free content that is still in use, producing a *freeze frame in Program* while the new scene or picture loads from the data drive or Graphic Hub.

To disable automatic superchannel scene cleanup

- On *every* Viz Engine in the ALP setup, set the following in the engine configuration (`VizEngine-*.cfg`):

```
SuperChannelSceneAutoFree = 0
```

The above setting keeps superchannel content in memory, so background loading does not disturb the program output.

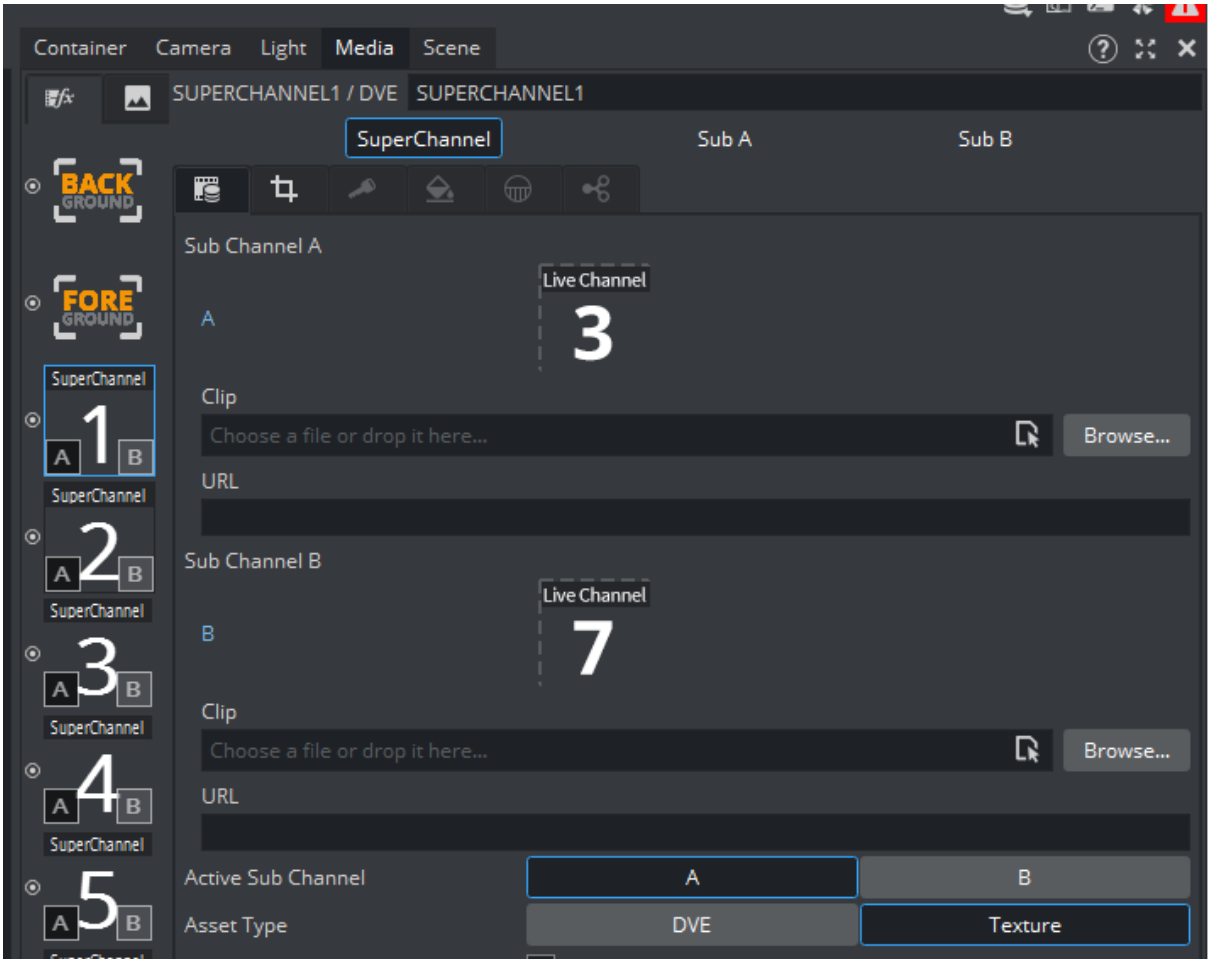
> **Info**: Scenes must be fully initialized before background loading.
> Together with `SuperChannelSceneAutoFree = 0`, this ensures graphics and pictures can be pre-loaded on the background/preview sub-channel without freeze frames in Program.

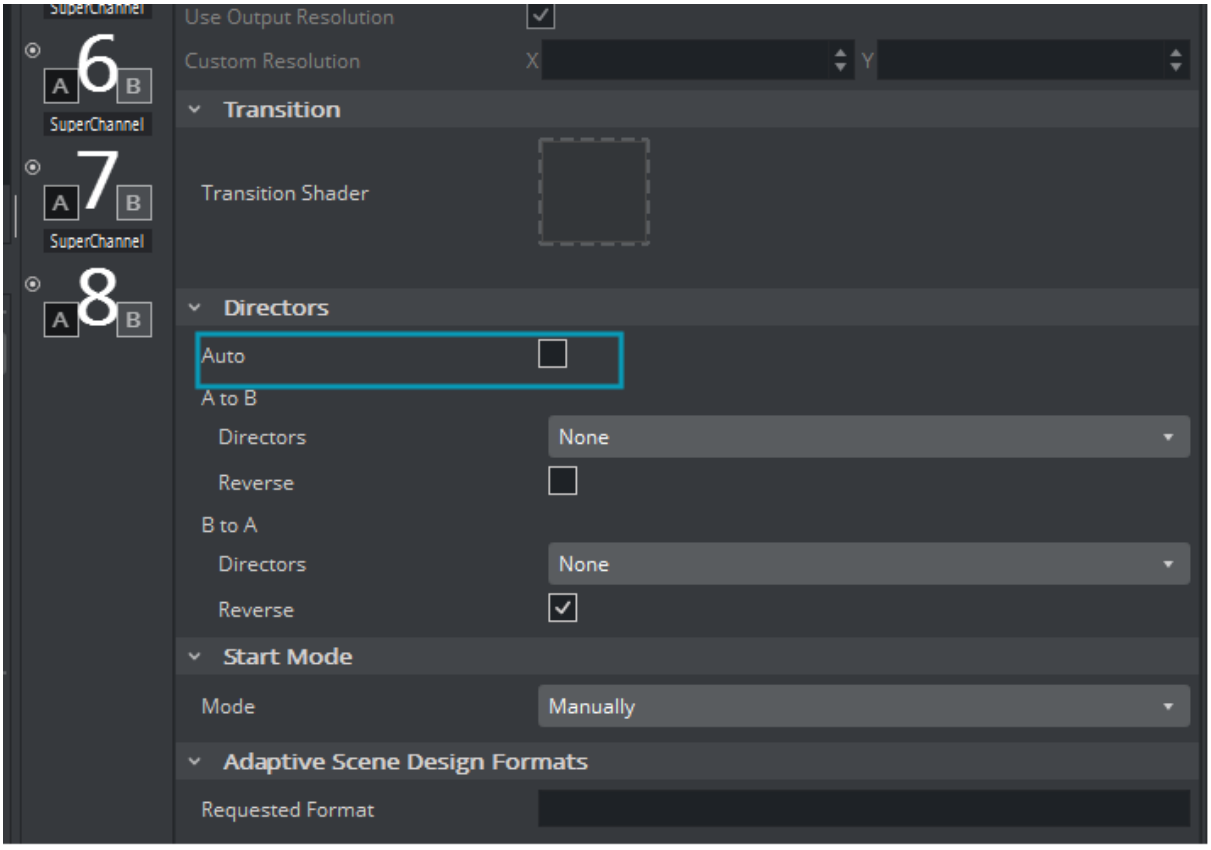
Viz Engine Scenes

Design or import the Viz Artist scenes that will be used for the engines in the Adaptive Live Producer setup. Each scene defines a *set* of layouts or presets (*1Box*, *2Box*, 3Box, *BigSmall*, and so on) with superchannels for live sources and graphics.

Requirements per scene

- Same number of superchannels across all engines (PP, ME1, ME2)
- Same preset names on all engines
- Content transition directors configured in the scene design (for example crossfade, dissolve, bookflip). These directors drive animated transitions between content on a superchannel.
- The *Auto* property on the *Directors* of each superchannel must be *off* to allow Mosart control the content transitions.

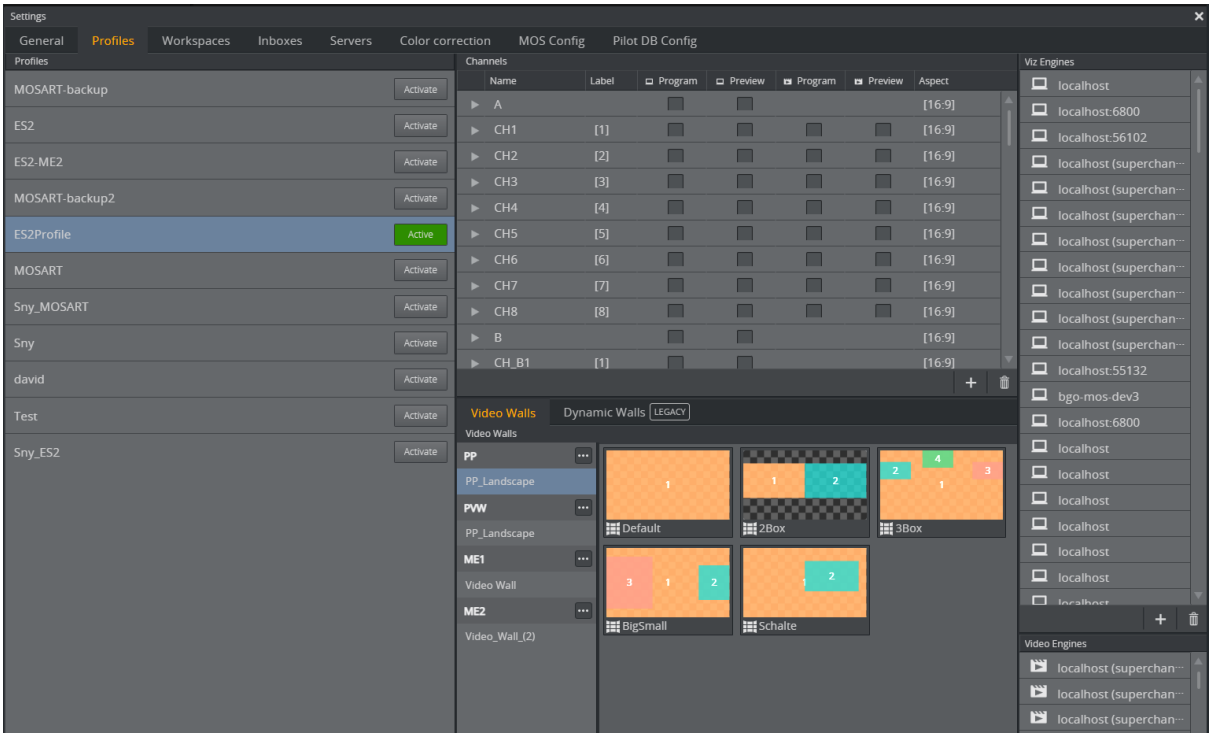




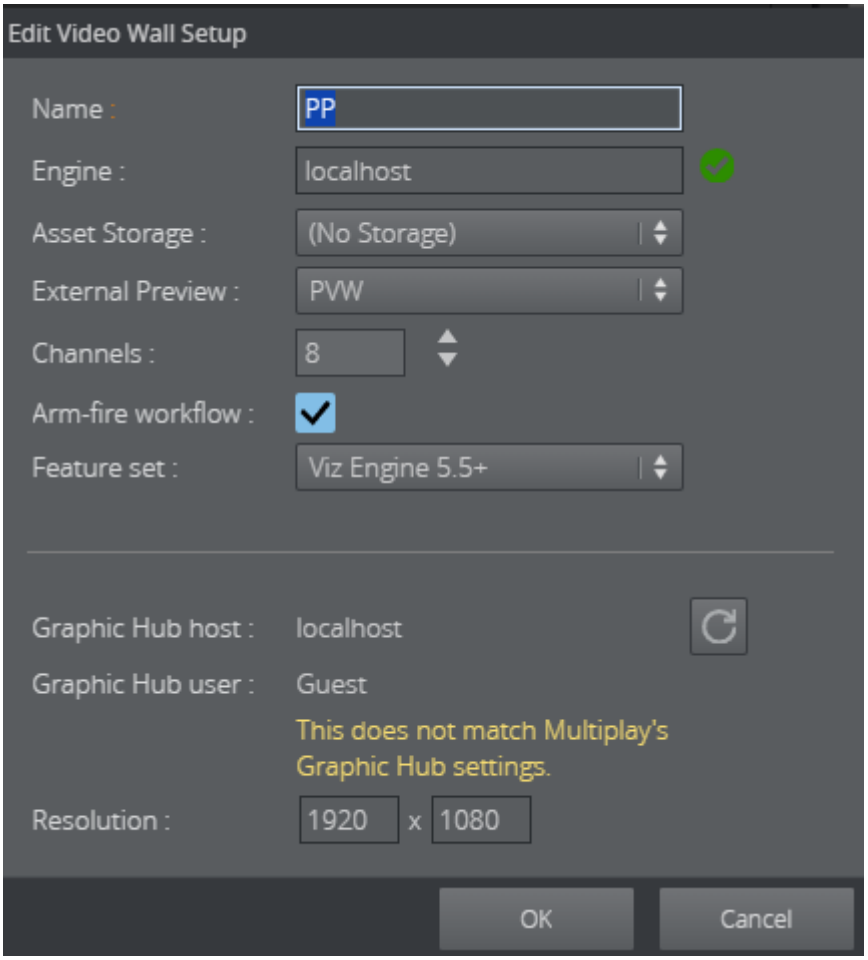
Mixer Effect Setup

Each Mixer Effect is configured with Viz Multiplay's *Profiles* configuration, for now. In the Viz Multiplay profile configuration, you set up a Mixer Effect as a **video wall**, the Viz Multiplay object that controls a Viz Engine's superchannels. The video wall terminology belongs to Viz Multiplay; in ALP, one video wall is one Mixer Effect.

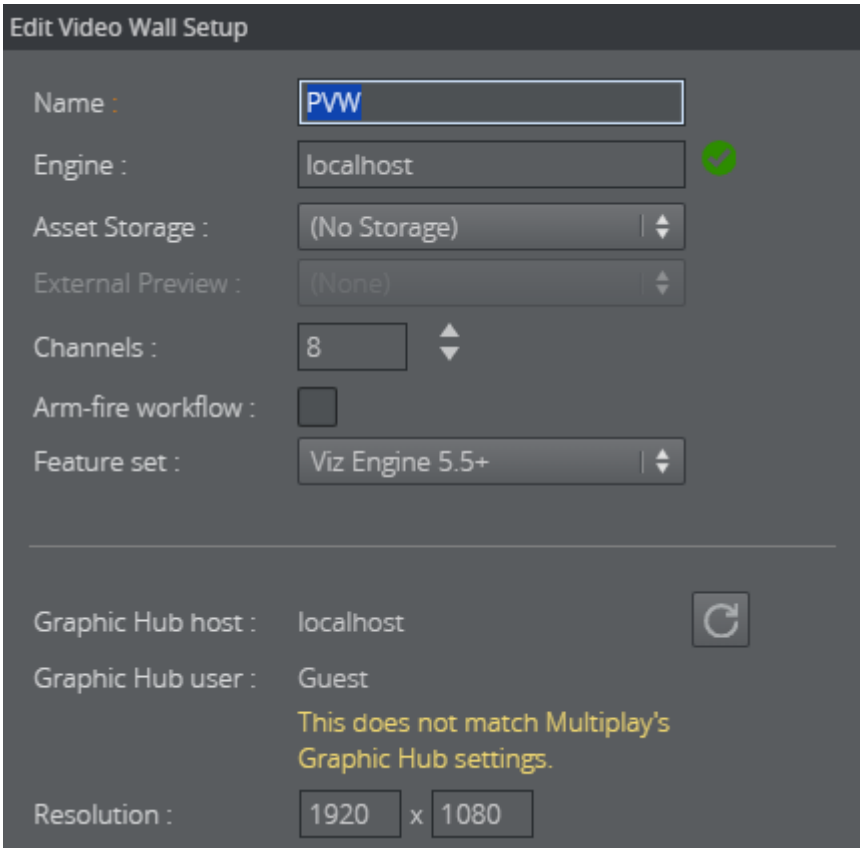
1. Create a **profile** with connections to each Viz Engine.
2. Create **video walls** for the main output (PP) and any secondary outputs (ME1, ME2, and so on).
   Each video wall corresponds to a Mixer Effect in Mosart (PP, ME1, ME2).
   Name the video walls as you desire (it does not have to be PP, ME1 etc.)



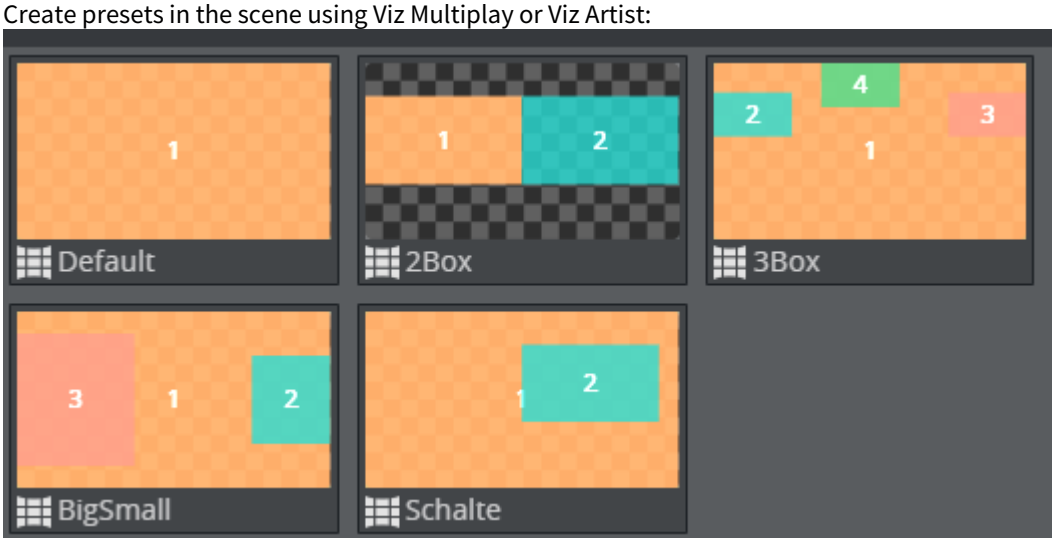
3. Verify that **Arm-fire workflow** is enabled and configure the required number of superchannels.



4. For each video wall, optionally configure a *preview wall* for preview output on a separate engine and link to it from the program wall.
   For the preview wall, do *not* select **Arm-fire workflow** if the preview is configured on the same machine as the main output. See details in section Preview Setup.



5. Create *scenes* for each video wall. Set the same scene in the matching preview wall.
6. Create presets in the scene using Viz Multiplay or Viz Artist:



7. Note down the *profile*, *video wall* and *scene names* from Viz Multiplay to configure them later in Viz Mosart. Mosart creates the show and playlist structure on the MSE during initialization, so there is no need to create them beforehand with Viz Multiplay.

Preview Setup

Preview is *optional*. When a preview wall is configured, Viz Mosart cues the next look to a separate engine so operators can see it before taking to program.

How preview behaves depends on where the preview engine runs.

Info:

- The *Arm-fire* mechanism enables *transitions in Preview.*
- Viz Mosart applies cue-time preset and content transitions through the arm-fire workflow.
- A preview wall can use arm-fire only when its engine runs on a *separate* machine from the program engine.

To define Preview on a separate machine (recommended)

1. Run the preview engine on a different machine from the program engine.
2. In Viz Multiplay, configure the preview wall as a full video wall in *arm-fire mode*, with the same scene and presets as its program wall, and select it as the preview wall of the program video wall.
3. With arm-fire on both walls, Mosart replicates the cue-time preset and content transitions on preview, so preview matches what program will show.

To define Preview as a second instance on the same machine

When the preview engine is a second Viz Engine instance on the same machine as the program engine, *arm-fire mode does not work for preview*. Preview still shows the cued content, but the transitions applied when cueing a template cannot be replicated on preview - preview uses a plain take with the scene's default transition.

1. Copy `C:\ProgramData\vizrt\VizEngine\VizEngine-0.cfg` and name the copy `VizEngine-1.cfg`.
2. Create a second shortcut for Viz Engine (or Viz Artist) that starts this second instance, then start both instances on the machine.
3. In Viz Multiplay, create two video walls for ALP pointing to the same scene — one for program and one for preview.
4. On the **Program** video wall, check **arm-fire workflow** and select the preview video wall as its preview.
5. On the **Preview** video wall, do *not check* **arm-fire workflow**.
6. In VDOM Abbreviations and key concepts (the Media Sequencer's data tree, browsable from the MSE home page at *http://mse-server:8580/app/vdomconfig/vdomconfig.html*) find the main channel for the preview video wall and copy its handler name (for example `viz_background_aad`).
7. In the scheduler, find that handler and set the main port for the second engine instance (for example `6800` instead of `6100`). Click **Apply**.
8. On the following handlers, correct the ports for the superchannels of the second instance (for example starting at `56201`). **Apply to all**.

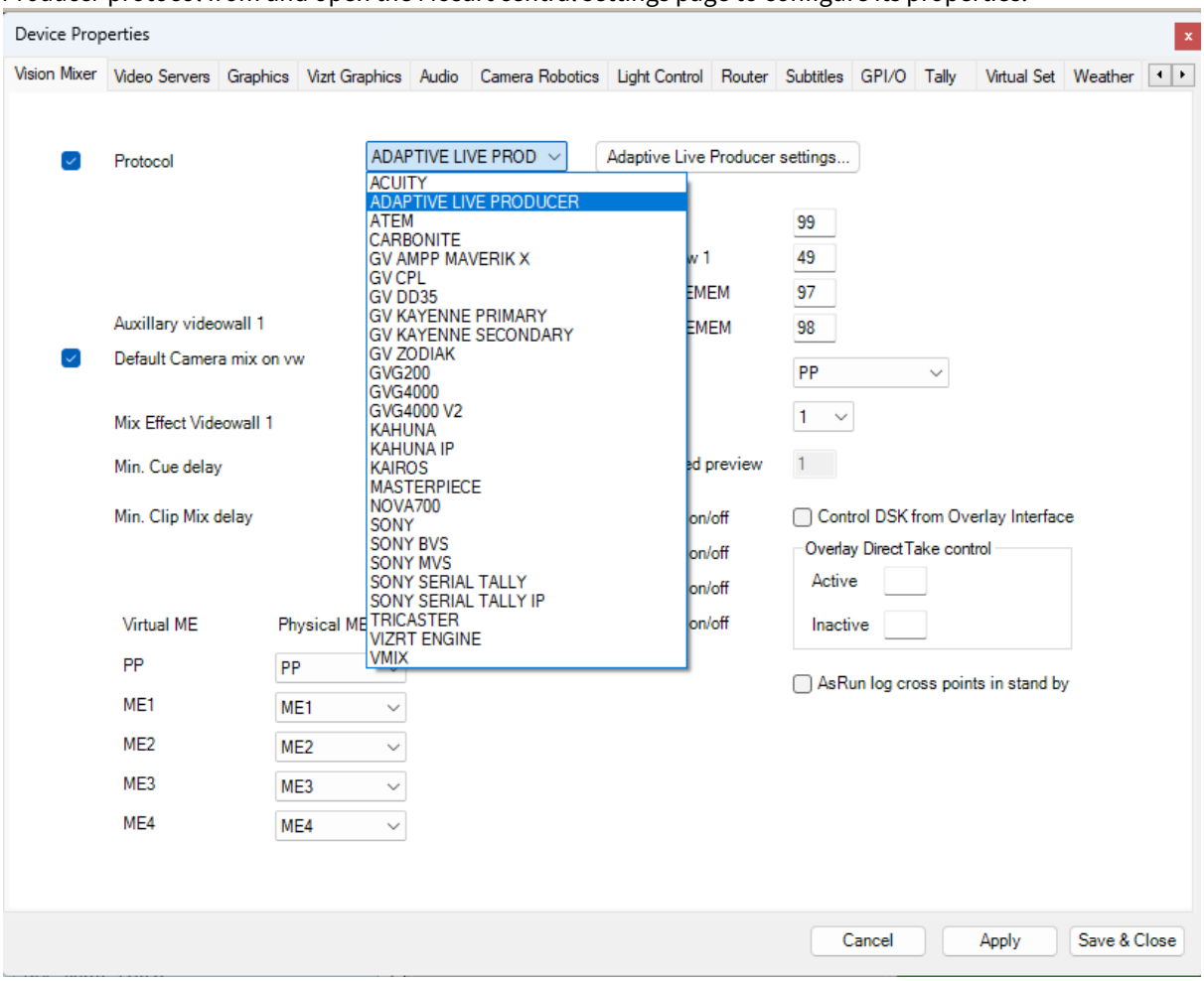
Note: If the preview video wall is later modified from Viz Multiplay, these manual port changes are overwritten with the defaults and you must *redo* the steps above.

Viz Mosart Setup

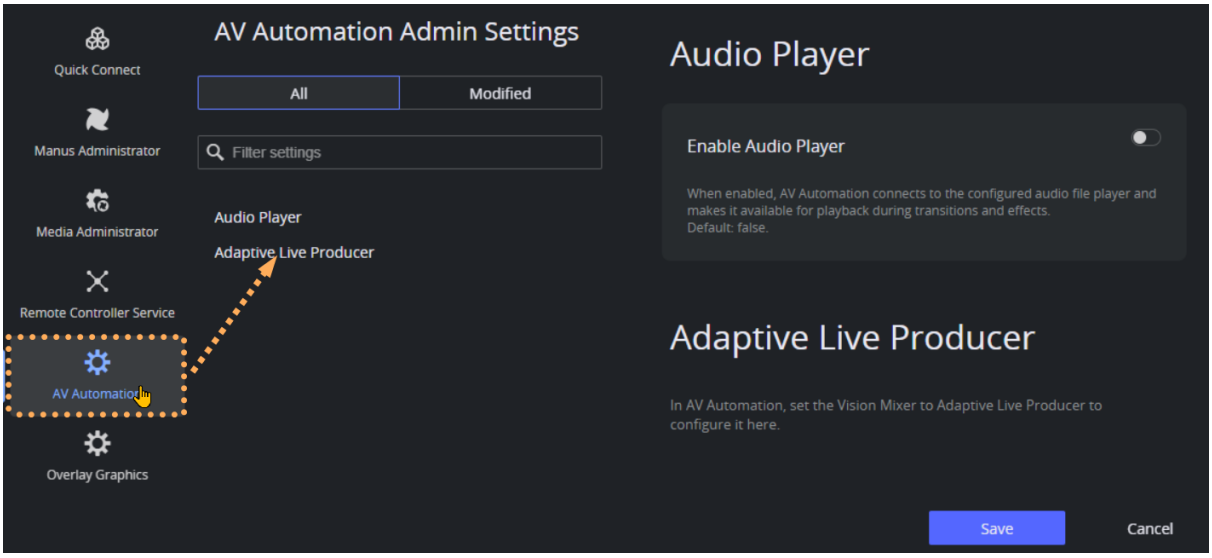
Device setup for Adaptive Live Producer

To configure ALP in Viz Mosart

1. In AV Automation navigate to **Devices > Properties > Vision Mixer** [tab] select and enable *Adaptive Live Producer* protocol from and open the Mosart central settings page to configure its properties.



2. In the AV Automation navigate to **Devices > Settings.**
3. In the combined **Settings** window that opens, select **Adaptive Live Producer**.



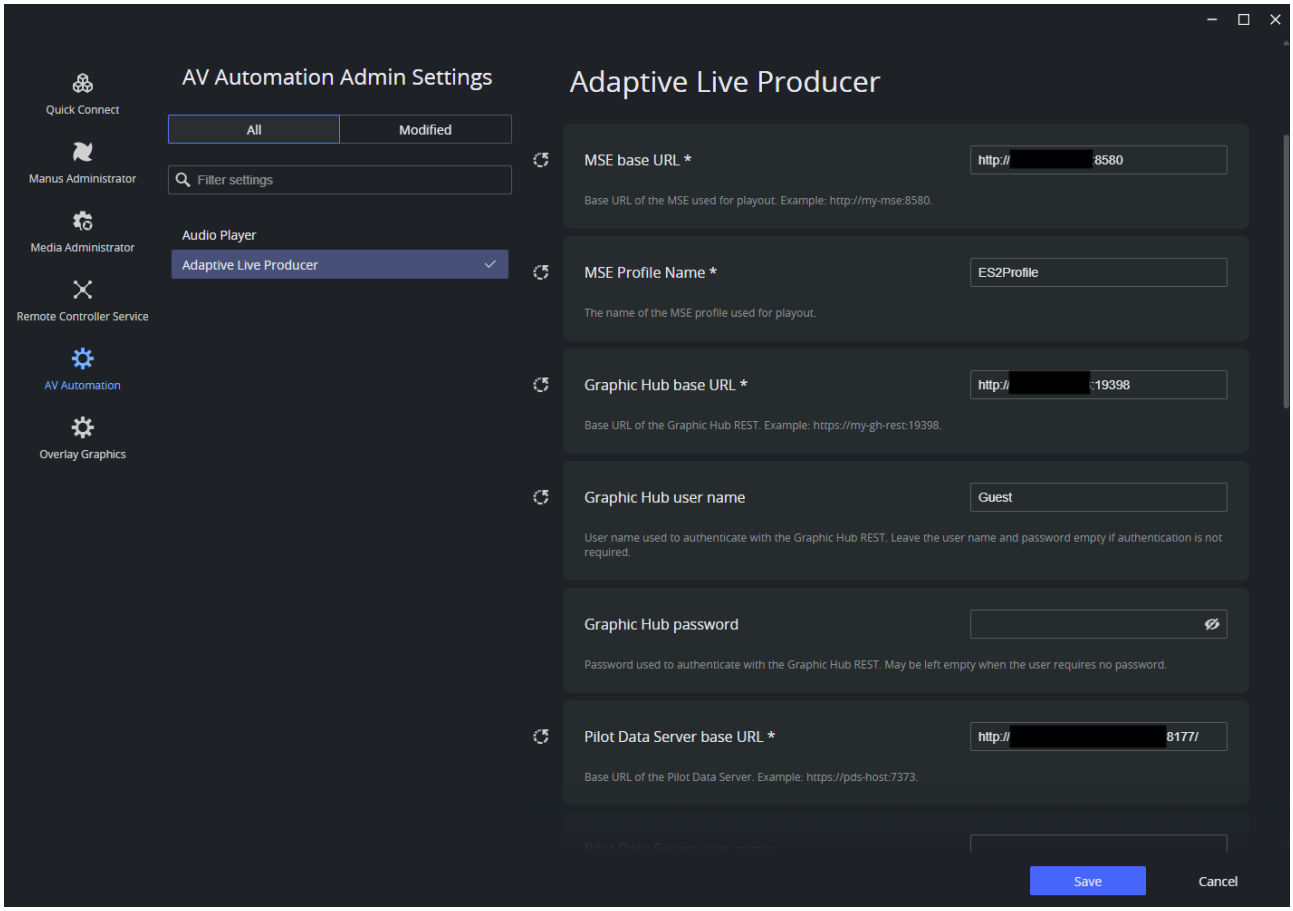
4. Configure the settings following these guiding values:

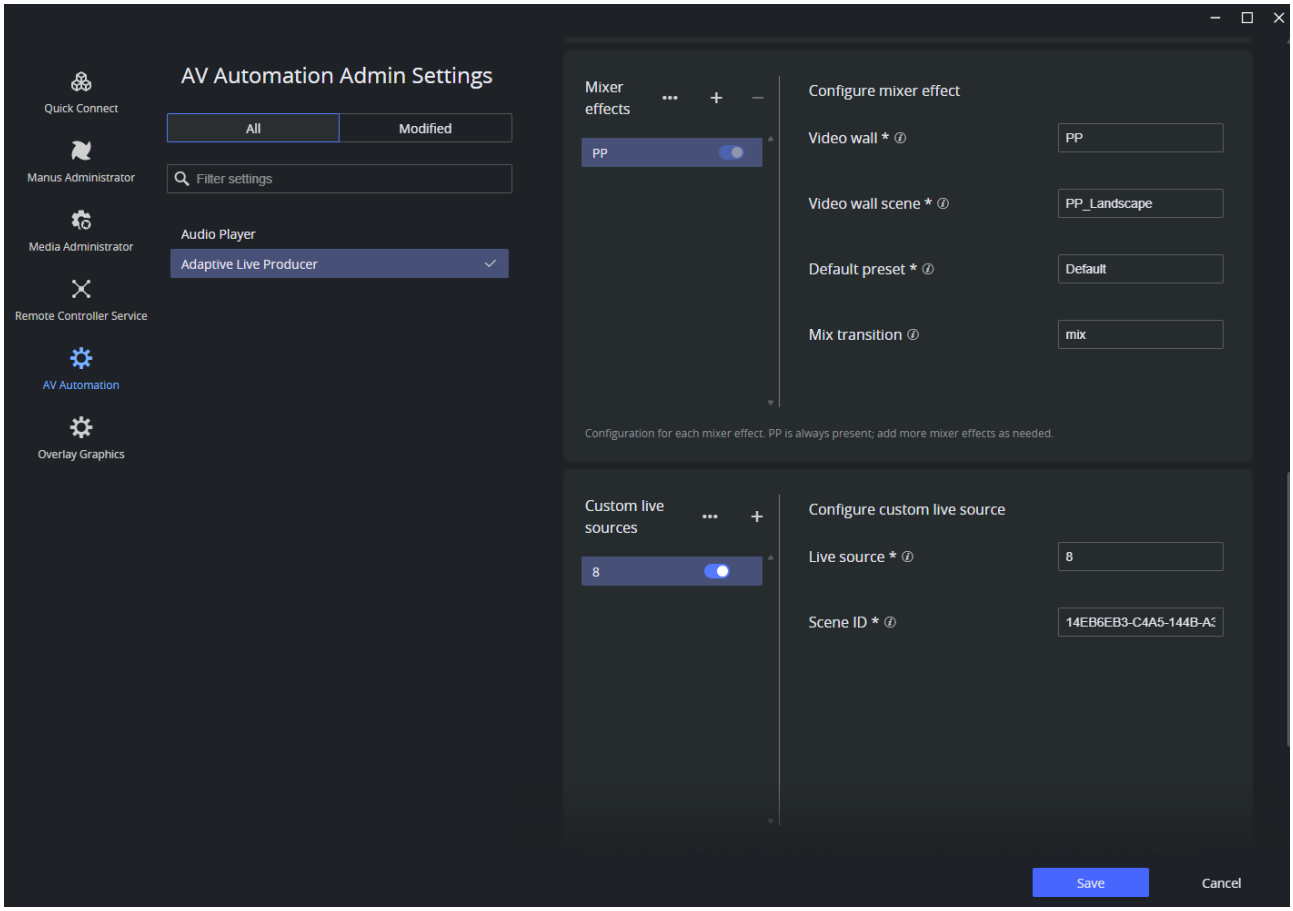
Setting	Scope	Description
MSE base URL	Global · required	Base URL of the MSE used for playout. Example: `http://my-mse:8580`.
MSE Profile Name	Global · required	Name of the MSE profile used for playout (default `MOSART`).
Graphic Hub base URL	Global · required	Base URL of the Graphic Hub REST. Example: `https://my-gh-rest:19398`.
Graphic Hub user name	Global · optional	User name to authenticate with the Graphic Hub REST. Leave the username and password empty if no authentication is required.
Graphic Hub password	Global · optional	Password to authenticate with the Graphic Hub REST. May be left empty when the user requires no password.
Pilot Data Server base URL	Global · required	Base URL of the Pilot Data Server. Example: `https://pds-host:7373`.
Pilot Data Server user name	Global · optional	User name to authenticate with the Pilot Data Server. Leave the username and password empty if no authentication is required.

Setting	Scope	Description
Pilot Data Server password	Global · optional	Password to authenticate with the Pilot Data Server. May be left empty when the user requires no password.
Mixer effects	Per mixer effect · required	*PP* is always present; add more mixer effects as needed. The names are auto-generated: ME1, ME2 etc.
Video wall	Per mixer effect · required	Name or ID of the video wall used by this mixer effect (as created in Viz Multiplay).
Video wall scene	Per mixer effect · required	Name or ID of the video wall scene used by this mixer effect (as created in Viz Multiplay).
Default preset	Per mixer effect · required	Preset used as the default when no *Switcher Plugin* command is added on a Mosart template.
Default transition	Per mixer effect · optional	Transition name used for Mix and Wipe with rate *> 0* transitions on the background (SC1). If empty, the whatever transition is currently used on the scene is used.
Custom live sources	Optional	Use this when a live input needs more than a plain video texture, for example a branded frame, a border, a rotation, or any other treatment built into the scene design. Custom live sources are optional. Each override targets one live source by number and applies to *every* mixer effect (*PP*, *ME1*, *ME2*, and so on). You cannot set a different scene per mixer effect.
Live source	Custom live source · required	The live source number the override applies to (1 to the number of live sources).
Scene ID	Custom live source · required	UUID of the Graphic Hub scene to use instead of that live source (for example, for branding).

Mixer effects is an expandable list: PP is always present, add ME1, ME2, … as needed.

Custom live sources are an optional list that applies to every mixer effect.


Quick Connect
Manus Administrator
Media Administrator
Remote Controller Service
AV Automation
Overlay Graphics
AV Automation Admin Settings
All
Modified
Filter settings
Audio Player
Adaptive Live Producer
Adaptive Live Producer
MSE base URL *
http:// 8580
Base URL of the MSE used for playout. Example: http://my-mse:8580.
MSE Profile Name *
ES2Profile
The name of the MSE profile used for playout.
Graphic Hub base URL *
http:// :19398
Base URL of the Graphic Hub REST. Example: https://my-gh-rest:19398.
Graphic Hub user name
Guest
User name used to authenticate with the Graphic Hub REST. Leave the user name and password empty if authentication is not required.
Graphic Hub password
Password used to authenticate with the Graphic Hub REST. May be left empty when the user requires no password.
Pilot Data Server base URL *
http:// 8177/
Base URL of the Pilot Data Server. Example: https://pds-host:7373.
Save
Cancel

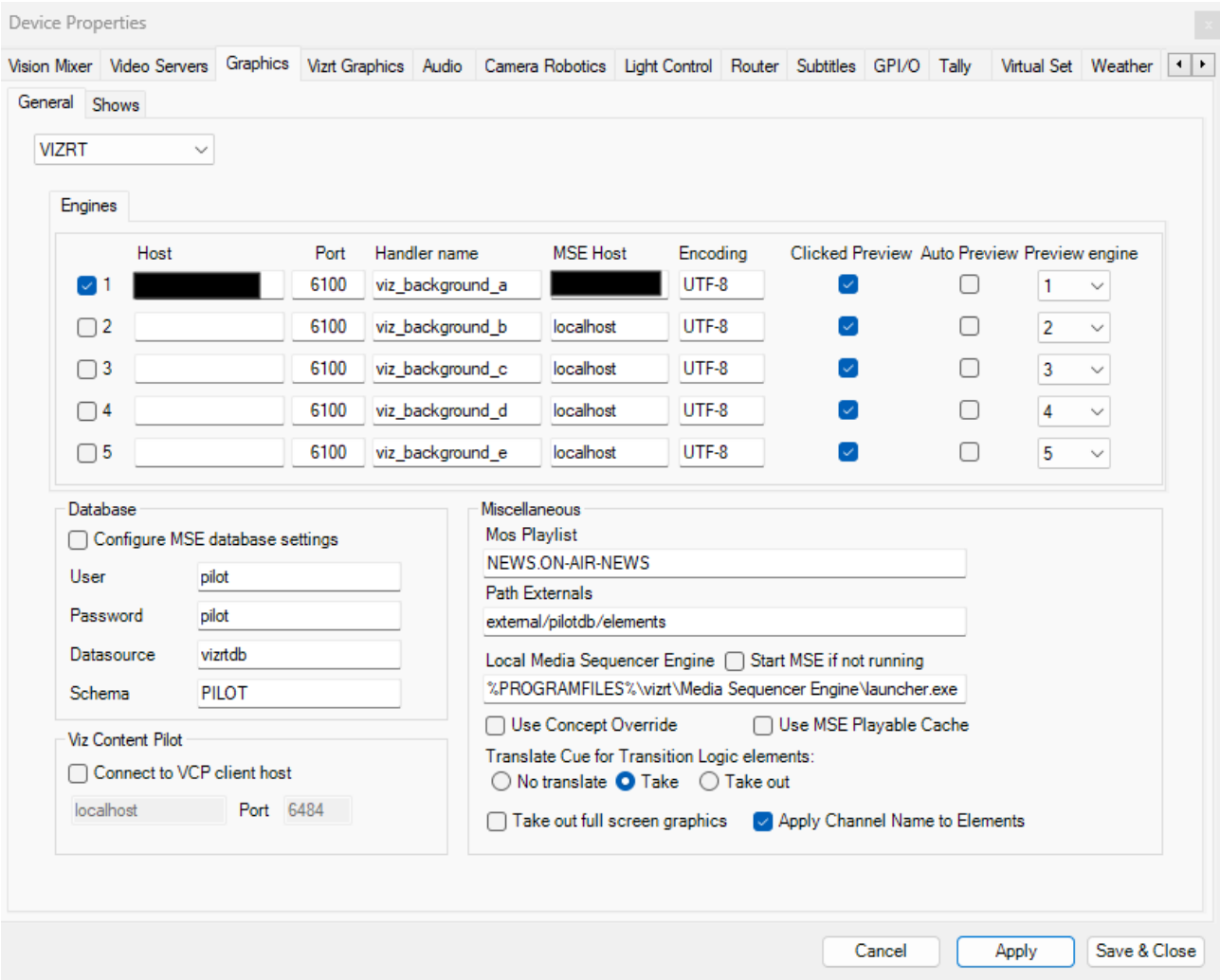


To set-up graphics for Adaptive Live Producer

Enabling the *Adaptive Live Producer* switcher in AV Automation also lets the graphics driver play out fullscreen graphics directly on the superchannels.

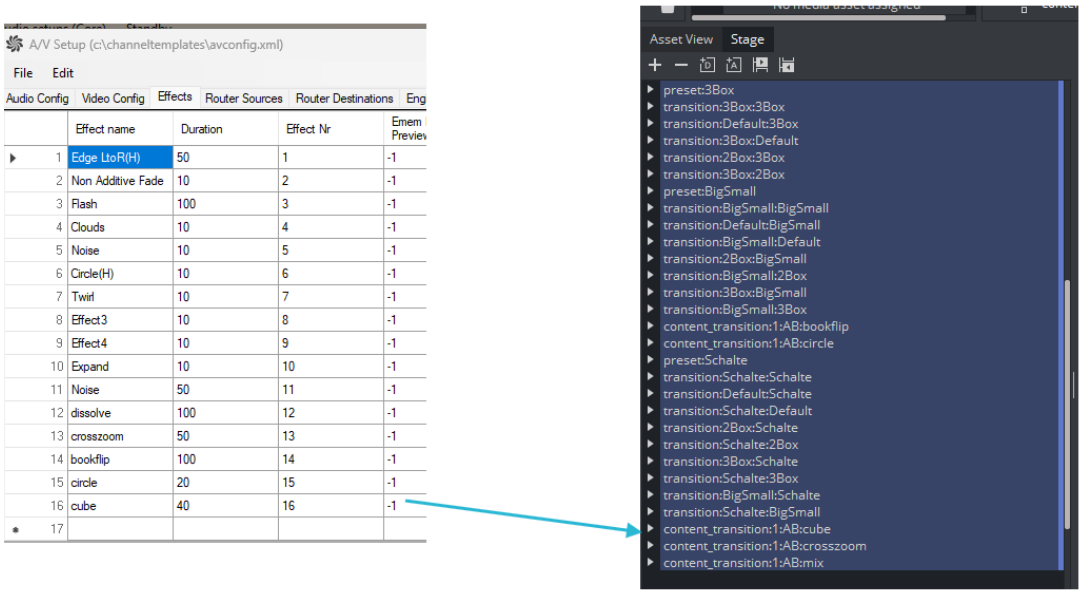
- In AV Automation navigate to **Devices > Properties > Graphics** [tab], follow the guidelines in the screenshot below.

> **Info**: In addition, the standard fullscreen graphics configured under **Device Properties > Graphics**[tab] keep working as before; their output is fed into ALP as a live input.



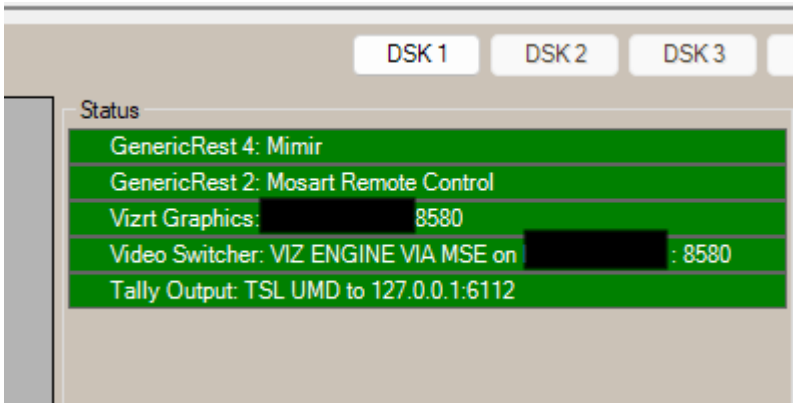
A/V Setup for Adaptive Live Producer

1. In AV Automation navigate to **Devices > Devices > A/V Setup** *and select a tab:*
2. Configure **Audio Config**
3. Configure **Video Config** sources.
4. Configure the **Engine Presets** list. Preset names must match those created in Viz Multiplay and Viz Artist.
5. Configure **Effects**. Each effect name must match a content transition *short name* in the Viz Engine preset scenes (for example *bookflip*, *dissolve*).
    a. All engines (PP, ME1, and so on) must use the same content transition names; the design of each transition can differ per engine.
    b. Both **custom** and **shader transitions** are supported.



Verify the Connection

At this stage you can confirm only the connection. In the AV Automation status panel, the ALP (vision mixer) connection should show as connected, reflecting the connection to the Media Sequencer.



A full functional test (cue and take a camera with the default preset, transitions, a DVE with keyfills, and fullscreen graphics) needs templates.

Build those first, then run the end-to-end checks described on *Templates and Show Design* section *Verify the full chain*.

Templates and Show Design

- Show design
  - Synchronization behavior
  - Preset and content synchronization rules
  - Preset handling
- Building templates
  - Controlling the background content
  - Composing a DVE layout box
  - Controlling the screen layout
  - To create a Switcher Plugin device command
  - Graphics
  - To control graphics content transition
  - Animating a fullscreen graphic out
  - Continuing a fullscreen graphic
  - Multiple graphics on DVE box layout
  - Graphics concepts (via macros)
  - Transitions
- Keyboard Shortcuts
- Control commands on templates
- Verify the Full Chain

This page covers how to build Mosart templates for Adaptive Live Producer (ALP) and how presets and content stay in sync on cue and take.

Show design

On a hardware switcher, taking a source to program is one self-contained action. With ALP, every take is composed in Viz Engine: the layout (preset) and the content on each superchannel are separate elements that the Media Sequencer arms and fires together. To design a show that behaves predictably, you need to know how Mosart keeps presets and content in sync, when it resets to a default preset, and what each template type does on take. This section covers those rules; read it before building templates.

> The Media Sequencer arm, fire, fire all, take etc. terms used in this section are defined in Abbreviations and key concepts on the Adaptive Live Producer page.

Synchronization behavior

When cueing a template, Mosart arms the preset and content for program, and takes the preset and content on preview (if configured). The preview take can carry the same transition as program when the preview wall is in arm and fire mode; otherwise it applies a fallback transition. Synchronized preset and content switching on preview, where everything updates in one frame, is **not supported**.

Synchronization between preset and content depends on how the action is triggered:

- **Cue-then-take (rundown flow)**: Mosart arms the preset and all related content. On take, it runs a fire-all so everything updates in the same frame.

- **Direct take or keyboard shortcut (no prior cue)**: Mosart performs a direct take only the affected content or preset. No preset and content synchronization.

Preset and content synchronization rules

Synchronization happens when a template is **cued** (rundown flow) with a *Switcher Crosspoint* or *KeyFills* command(s), together with a preset (default or custom via *Switcher Plugin*). On take, the preset, the background/keyfill content, and any graphics on that template all fire together in the same frame. *Graphics* command never triggers synchronization on its own, and never disable it. A **direct take, keyboard shortcut on program,** or **crosspoint change from the Mosart UI** (no prior cue) is not synchronized; those elements take individually.

Template / action	Synchronize preset and content
Switcher Crosspoint + preset (default or custom via *Switcher Plugin*)	yes
KeyFills (no *Switcher Crosspoint*) + preset (default or custom via *Switcher Plugin*)	yes
Graphics together with a *Switcher Crosspoint* and/or *KeyFills* + preset (default or custom via *Switcher Plugin*)	yes (the crosspoint/keyfills drive the sync; graphics fire on the same fire-all)
Graphics only (no *Switcher Crosspoint*, no *KeyFills*)	no (graphics take on their own director)
Any control commands to set a source or preset	no
Directtakes, keyboard shortcuts, crosspoint change from Mosart UI	no
Accessories as secondaries on the main primary item	no

Note. A synchronized take is all-or-nothing. If any armed element (a *Switcher Crosspoint*, *Key*Fill, or the preset) is invalid, the entire take is held back, including the valid elements. Viz Mosart does not yet show this status in the UI, so check that every configured source and preset exists before taking the template on air.

Preset handling

- When a *Switcher Plugin* command defines one or more presets per mixer region, those presets are used instead of the defaults.
- When *no Switcher Plugin* is defined, Mosart uses the *default preset* per mixer region (configurable in the switcher settings).
- A *reset to the default preset* happens when the template has no *Switcher Plugin* **and** has a *Switcher Crosspoint*. The reset runs on both cue and direct take.

Copyright © 2026 Vizrt

- A *fullscreen graphics template with no Switcher Plugin* also applies the default preset, even without a *Switcher Crosspoint*, because a preset must be loaded before a superchannel can show the graphic. Add a *Switcher Plugin* set to the wanted preset to keep a specific layout.
- Templates with *KeyFills only* (no *Switcher Crosspoint*) do *not* trigger a preset reset. Only the keyer content is synchronized.

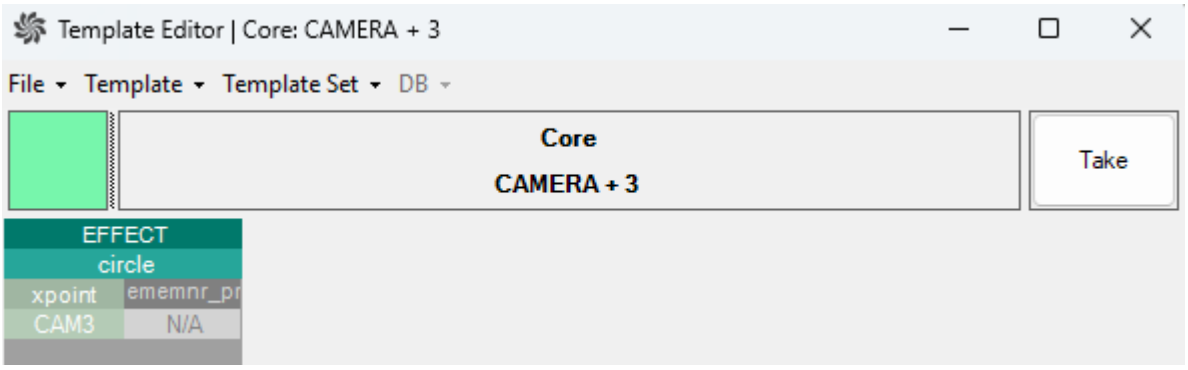
Building templates

Controlling the background content

A *Switcher Crosspoint* sets the background content (superchannel SC1) of a mixer effect. This is the standard way to change the background source. Point the crosspoint at the source you want on the background and select the mixer effect (PP, ME1, and so on).

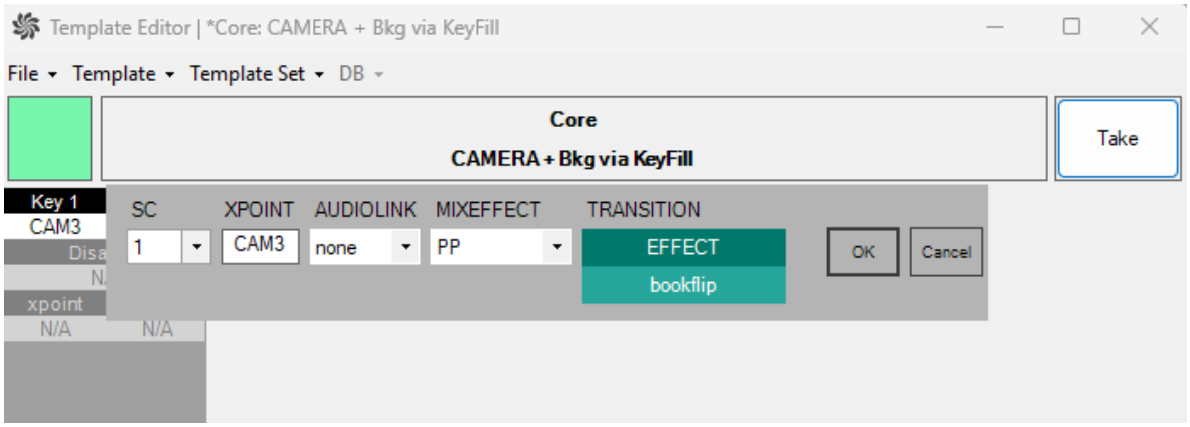
> **Info**: *SC1 is the background by convention, not necessarily by look.* The scene designed for a mixer effect can include its own main background, with the superchannels (SC1 and up) acting as keyers over it. Even then, a *Switcher Crosspoint* always targets SC1, so SC1 is the layer that background source changes land on. If your scene's real background is built into the scene rather than fed on SC1, keep in mind that the *Switcher Crosspoint* (and other background-targeting commands) still drive SC1.

The background take can be a hard cut or carry a transition: a Mix/Wipe with a rate, or a named effect from A/V Setup. Set the transition on the template or drive it from the NRCS. See Transitions for how each option behaves on air.



When a template has a *Switcher Crosspoint* but no *Switcher Plugin*, the preset also resets to the default (see Preset handling).

A **KeyFill** command can set the content for any superchannel and as such, the background content as well. Read more on the next section, Composing a DVE layout box.



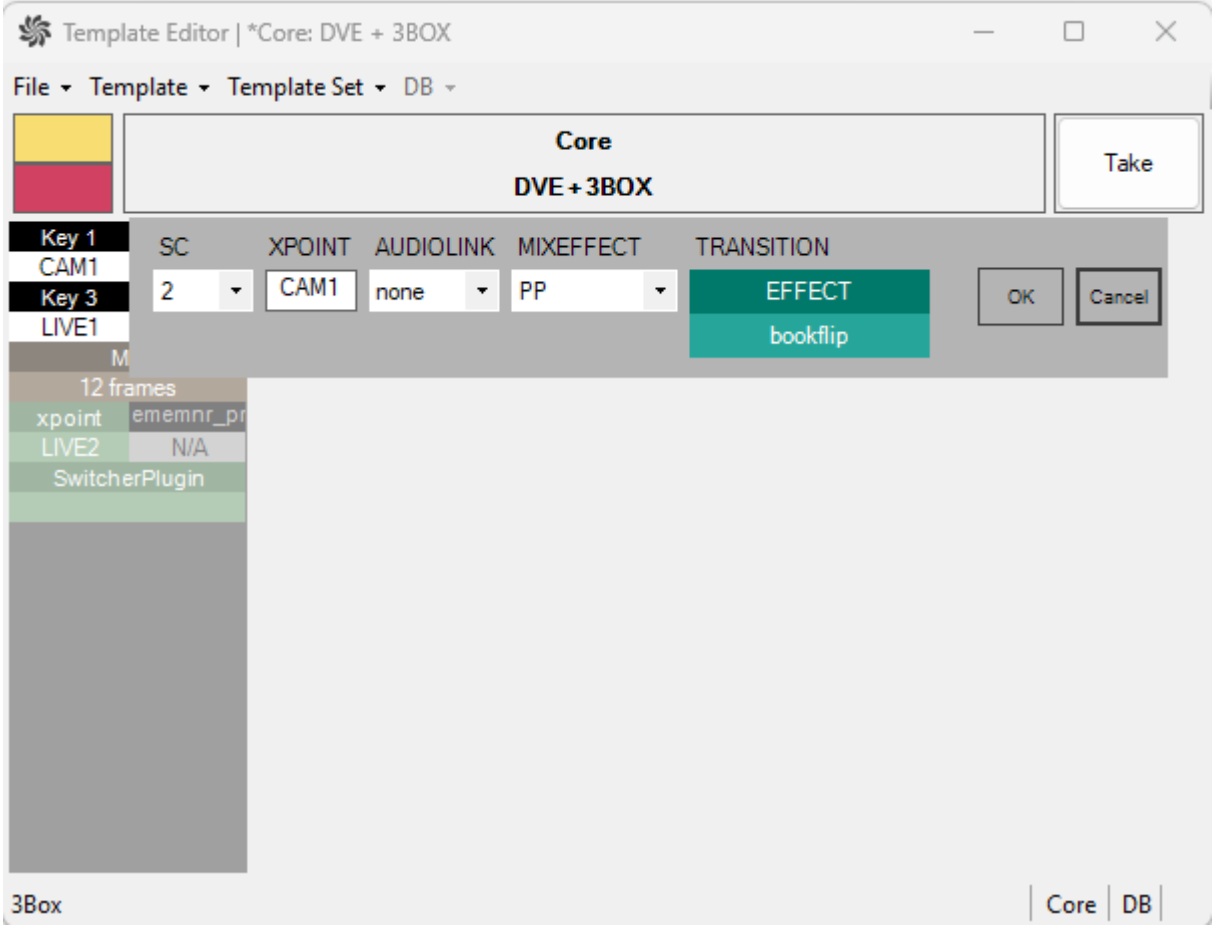
Composing a DVE layout box

A DVE needs two separate things, and Mosart sets them independently.

- A *preset* (1Box, 2Box, 3Box, and so on) defines the *layout* only: which superchannels are visible and where their boxes sit on screen.
- The *content* of each box is set per superchannel.
    - The background (*SC1*) from a *Switcher Crosspoint* (or a KeyFill).
    - Each box (*SC2* and up) from a *KeyFills* command.

A **KeyFills** command sets the content on the keyer superchannels (SC1 and up) of a mixer effect. Use this, for example, to build a DVE box layout directly on PP. For ALP, each *KeyFill* row can carry its own transition, set in the **TRANSITION** column of the KeyFills dialog: a Mix with a rate, or a named Effect from A/V Setup. See Transitions for how each option behaves on air.

For ALP you can now also use a *KeyFill* to set the content for the background (SC1).



Note. A KeyFill transition plays through the cue-then-take (rundown) flow. A direct or un-cued take of a KeyFill source uses a hard cut.

The property *KeyEnable* for a *KeyFill* which controls a keyer visibility on a traditional switcher is not available for ALP. In other words, the superchannel visibility can not be controlled with the *KeyEnable* property. Content on a superchannel is visible only when the active preset exposes that superchannel. A template with KeyFills on SC2 and SC3 but no preset shows nothing while the current preset is 1Box. Set the preset on the same template, or from a previous take, so the KeyFill content appears.

The opposite case matters too: a preset can expose more boxes than the template fills. A preset only reveals whatever content is already on the superchannels it shows, so a box you do not set keeps the content left there by an earlier take.

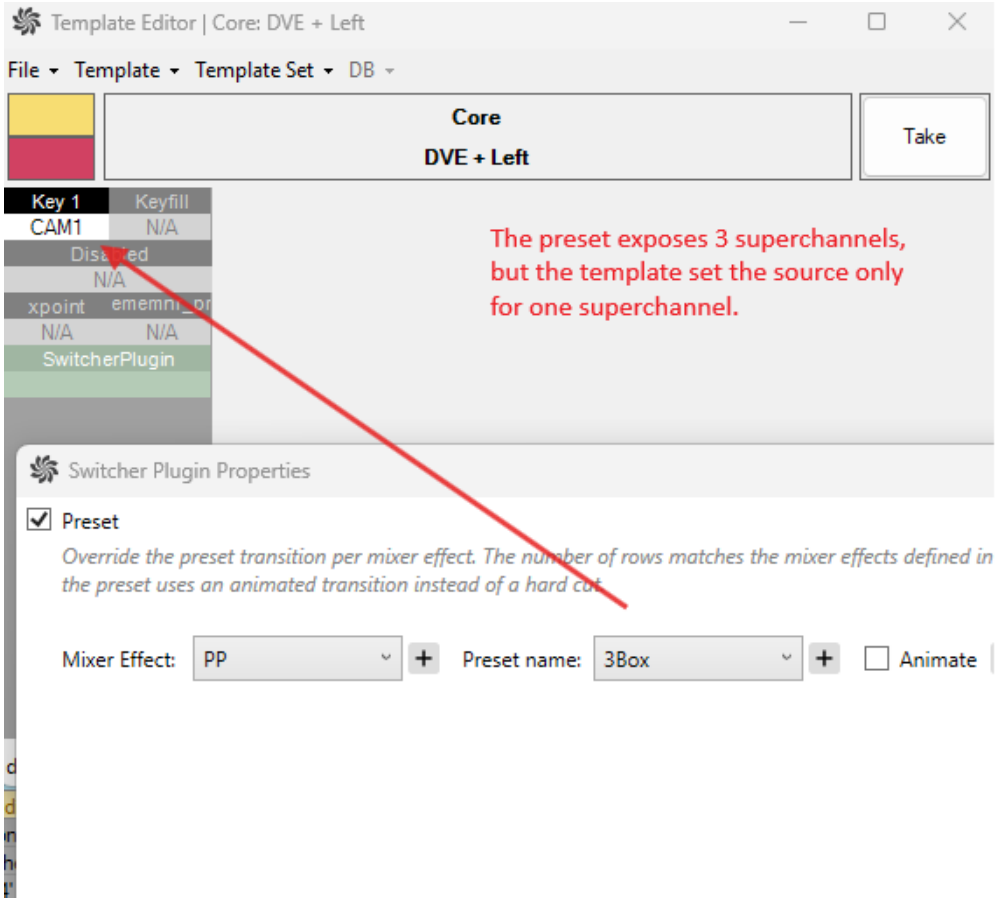
Note: *Fill every box that the preset exposes*.

- Any superchannel you leave unset keeps its previous content, and that old content goes on air.
- A template that takes a preset must set the content on every superchannel that preset shows.

Example

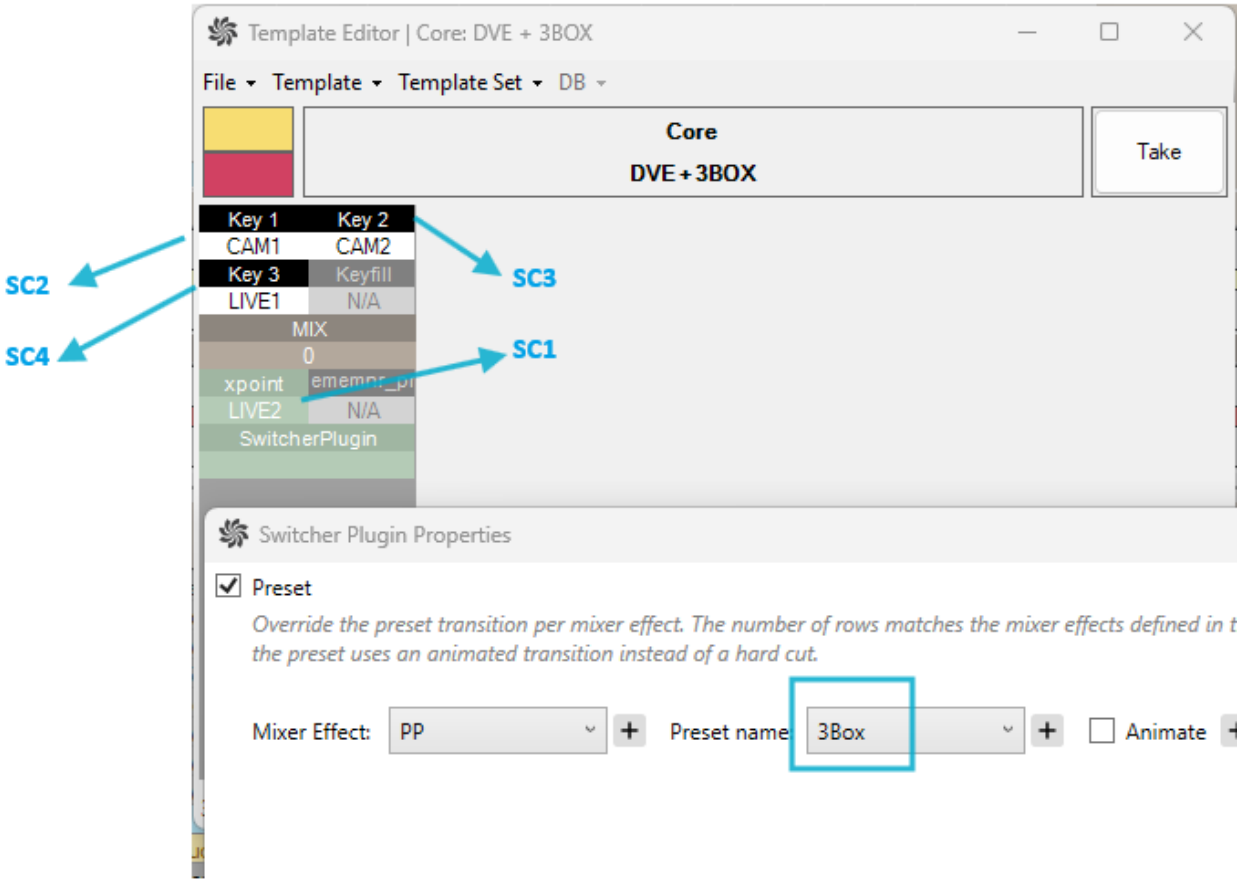
This preset shows the background plus three boxes:

Region in the layout	Superchannel
2, left box	SC2
3, right box	SC3
4, top box	SC4



A template that selects this preset but sets a KeyFill only on **SC2** (for example CAM1) leaves SC3 and SC4 untouched. On air, SC2 shows *CAM1*, while *SC3* and *SC4* still show whatever was *last set on them*, which may be stale content from a previous story.

- To get a predictable look, set content for SC1, SC2, SC3, and SC4 on the *same* template.
- In the Template Editor, an `N/A` on a KeyFill row means no content is set for that superchannel, so its box keeps its previous content.



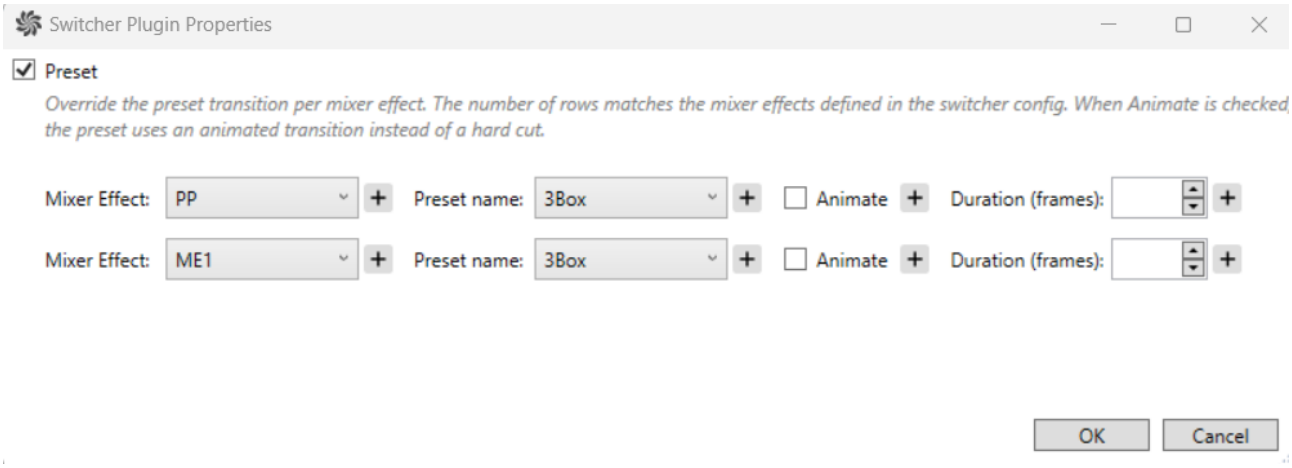
Controlling the screen layout

To select a preset for a template, add a **Switcher Plugin** device command.

To create a Switcher Plugin device command

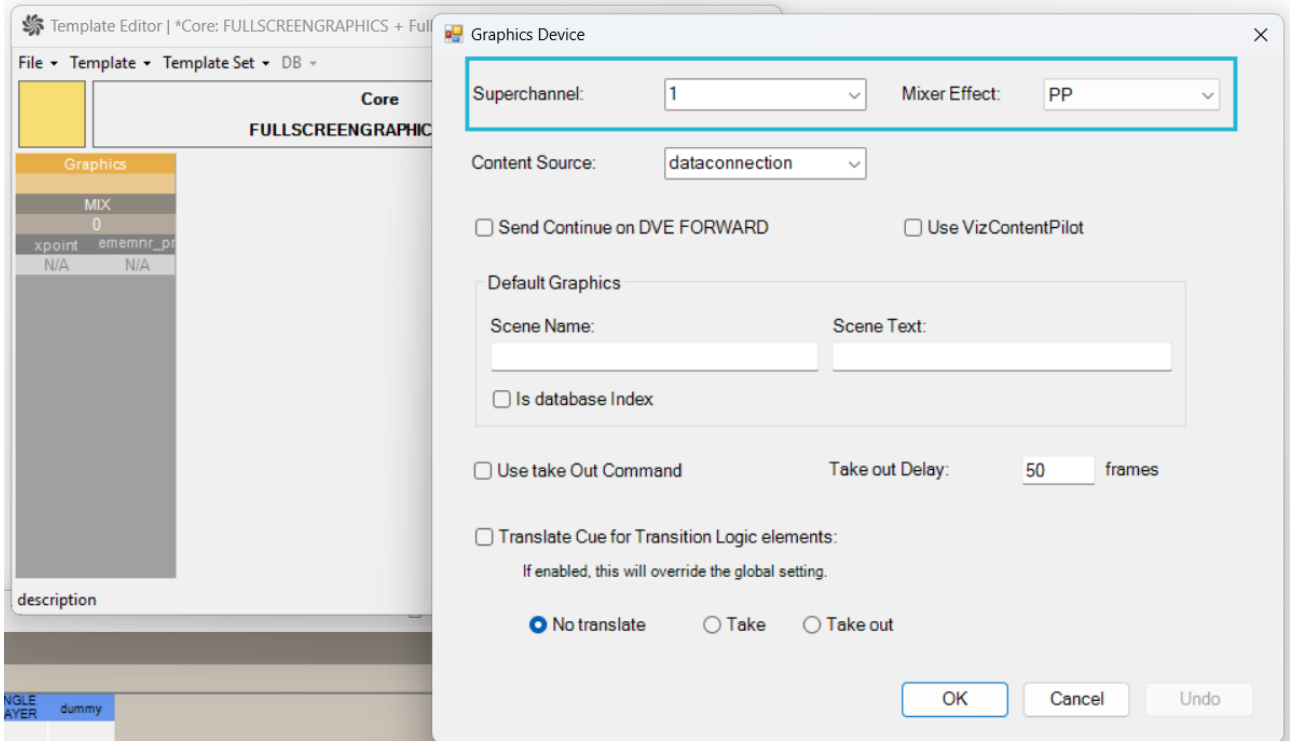
- Open the Switcher Plugin Properties dialog and configure:
  - **Preset**: Check to enable a preset override for this template.
  - **Mixer Effect**: Select the mixer effect (PP, ME1, and so on) corresponding to the Viz Engine instance where the preset will be set.
  - **Preset name**: Select the preset from the Engine Presets list (for example 2Box).
  - **Animate**: Check to make the preset use the animated transition as designed in the scene, instead of a hard cut.
  - **Duration (frames)**: Animation duration in frames (applies only when **Animate** is checked).

The number of rows matches the mixer effects defined in the driver config. Each row independently controls one mixer effect.



Graphics

To take graphics on any superchannel of an ALP, add a **Graphics** device command on the template. In the *Graphics Device* dialog, select the **Superchannel** and **Mixer Effect** where the graphic should appear. For a fullscreen graphic on the main output background, set Superchannel to *1* and Mixer Effect to *PP*. No *Switcher Crosspoint* is needed on the template (as required when working with a traditional switcher), only the **Graphics** device command.

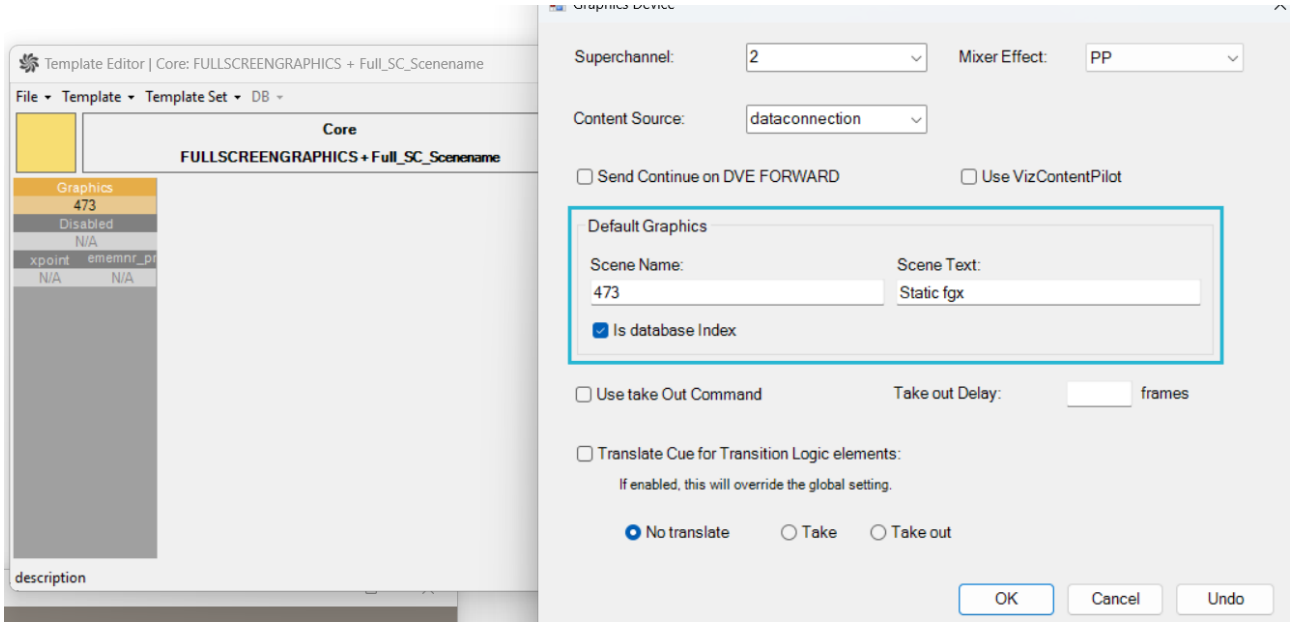


Graphic content can come from two sources:

- **NRCS**: A Pilot Edge MOS object placed in the story after this template. The graphic content loads from the newsroom system.

- **Static graphic**: Enter the graphic ID in **Scene Name** field and check: **Is database Index**.

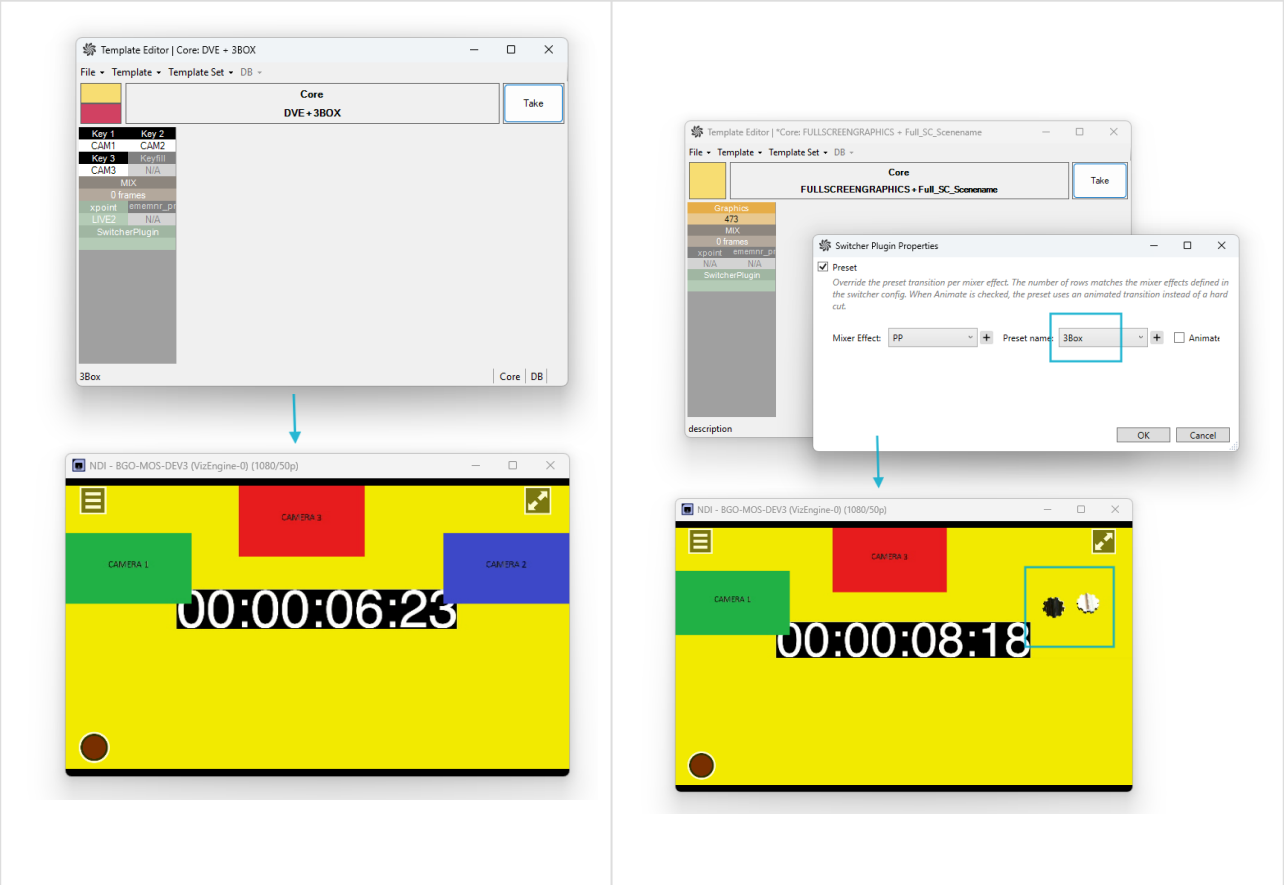
To add a graphic on the boxes of a DVE (for example 3Box), add a **Graphics** device command and select the *Superchannel* on the *Mixer Effect* where the graphic should appear. Provide the graphic ID in **Scene Name** with *Is database Index* checked, or provide content from the NRCS on a story containing this template followed by a fullscreen graphic MOS object.



Note: *Keep the layout when taking a graphic on a superchannel*.

- A graphics template with no *Switcher Plugin* applies the default preset configured in the switcher settings, because a preset must be loaded before a superchannel can show the graphic.
- If you are on air with a DVE layout such as 3Box and the next template takes a graphic on a superchannel, the layout *resets to the default* unless the template keeps the same preset.
- To take the graphic without changing the layout, add a **Switcher Plugin** to the template set to the same preset that is on air (*3Box* in this example).

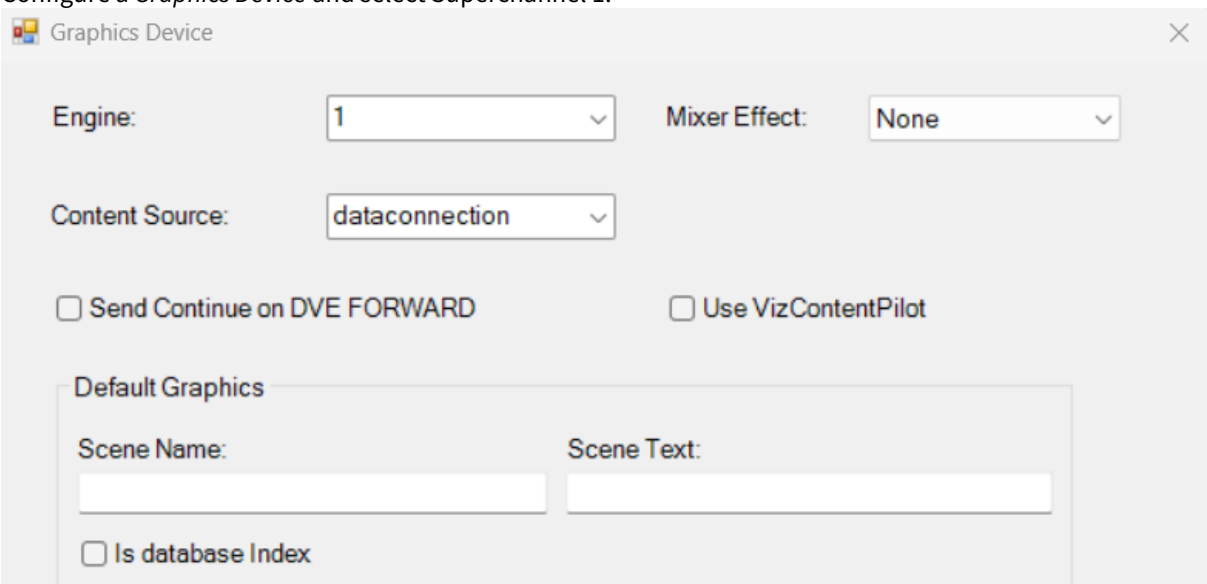
In the example below, a a 3Box layout is onair and next a graphic is taken on superchannel 3. To preserve the layout, a **Switcher Plugin** was added to set the same preset as currently onair.



To control graphics content transition

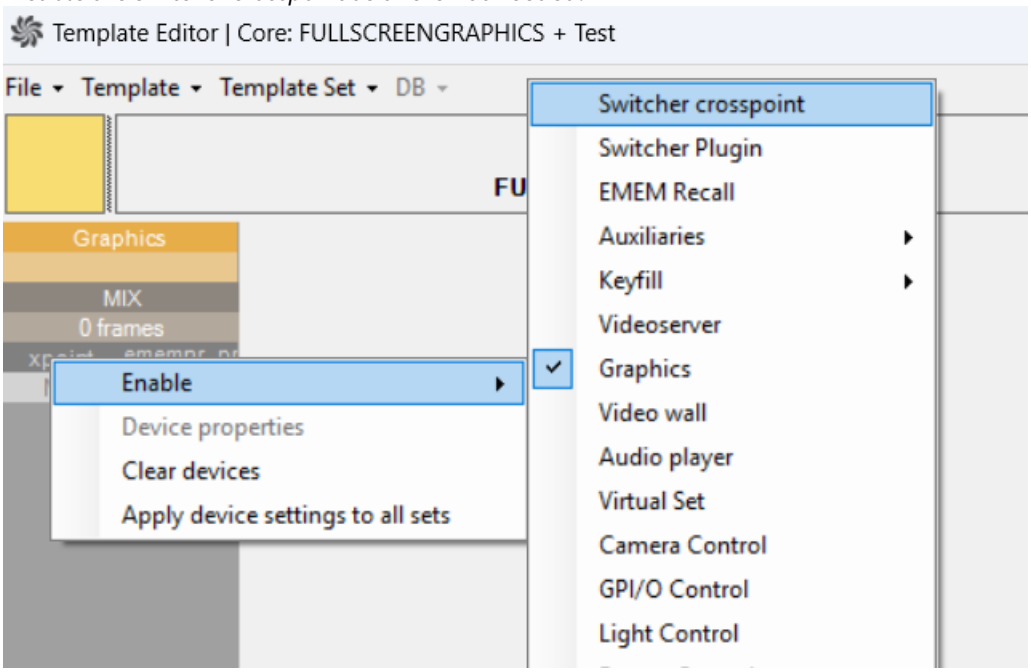
For now, only the content transition for superchannel 1 can be controlled from Mosart templates when taking graphics as content.

1. Configure a *Graphics Device* and select Superchannel 1.

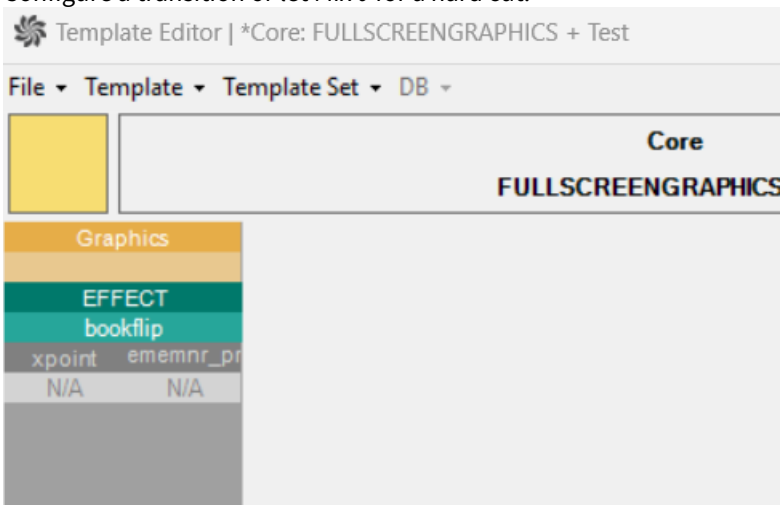


2. Add a *Switcher crosspoint* to enable the transition.

3. Disable the *Switcher crosspoint* as this is not needed.



4. Configure a transition or let Mix *0* for a hard cut.



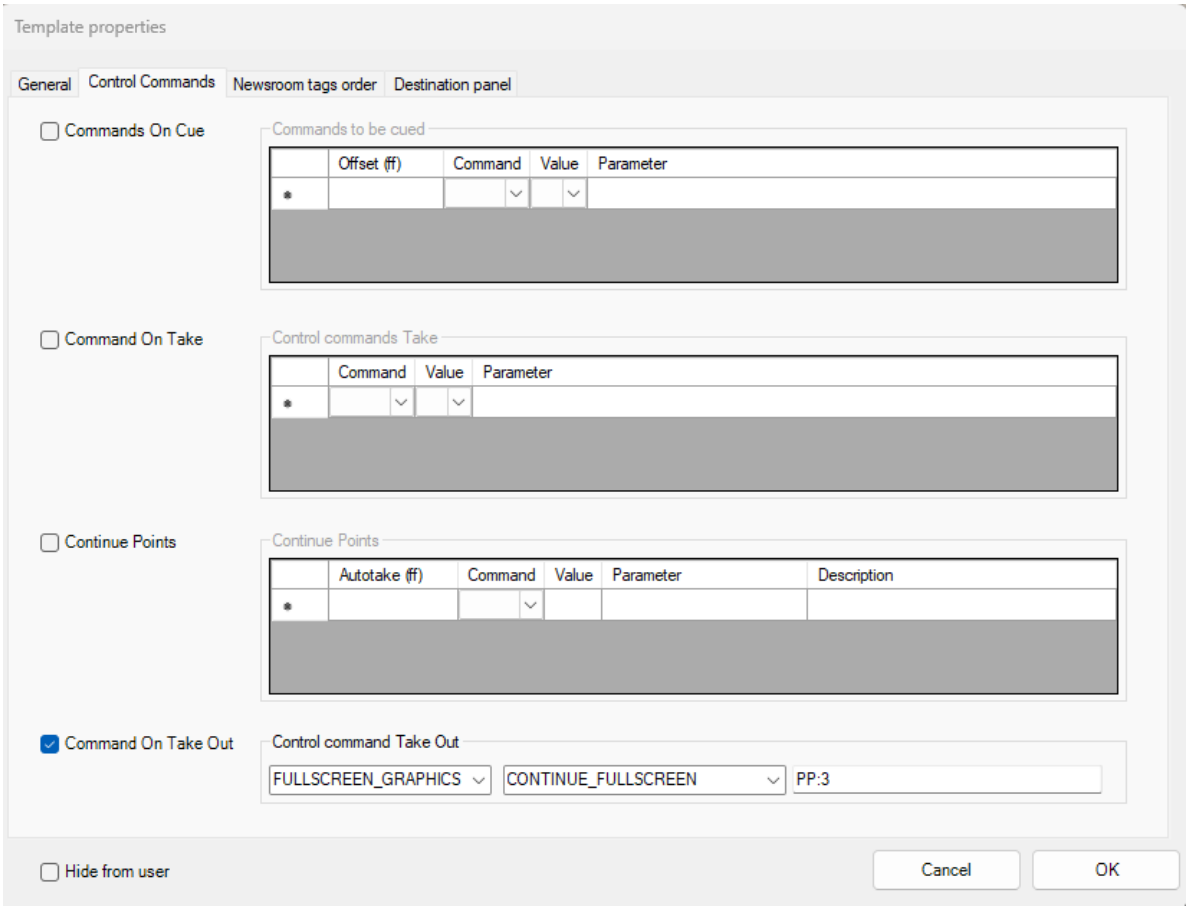
Animating a fullscreen graphic out

When you replace a fullscreen graphic with a live source, you usually want the graphic to play its out animation instead of just disappearing. Triggering an *out* animation depends on how the graphic scene is built.

Standalone graphics (out animation on a continue point)

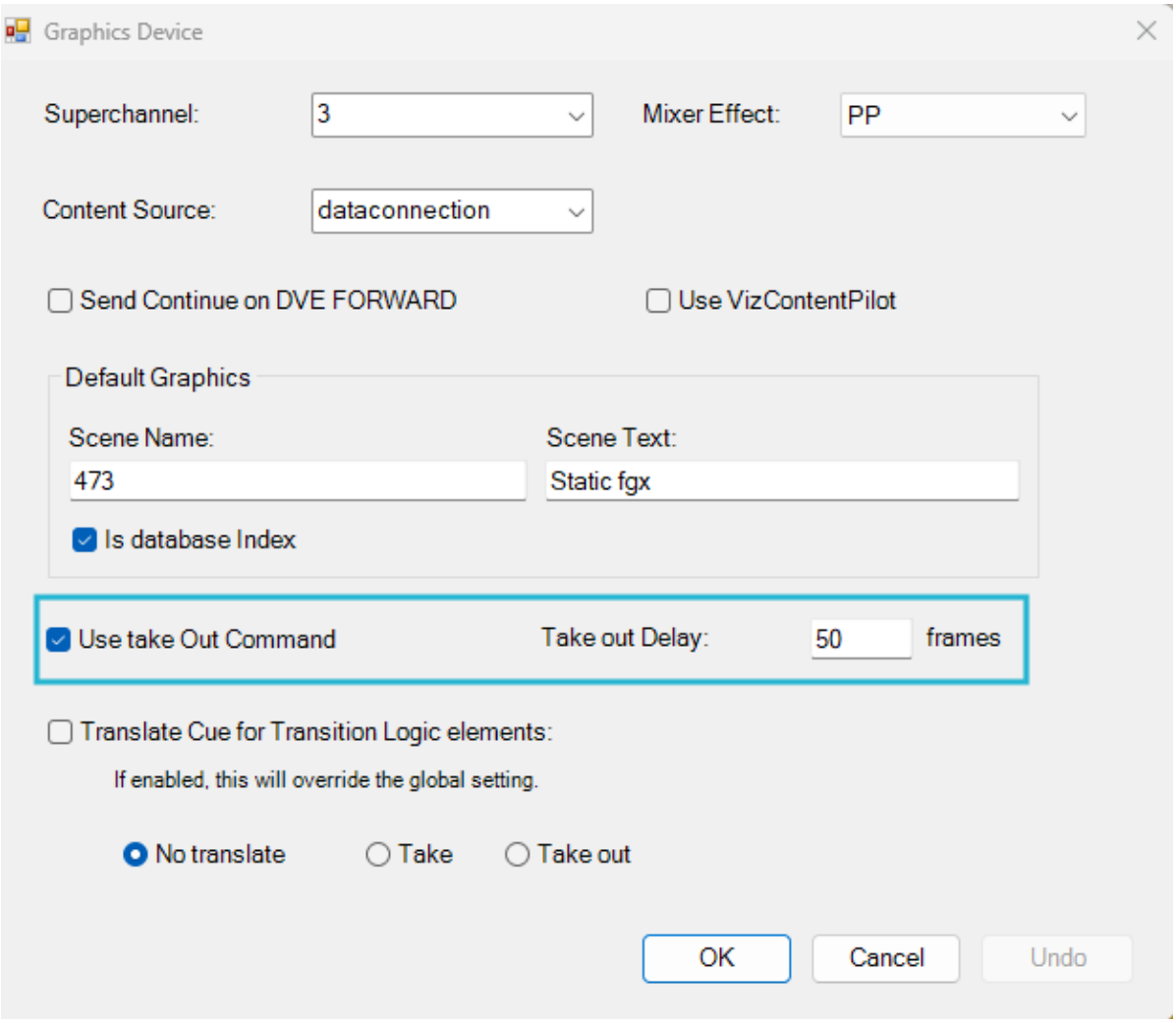
These graphics usually build their out animation behind a continue point in the scene. The *Take Out* option on a Graphics device command removes the graphic directly, it does not advance the scene to that continue point, so a take out skips the out animation. To play the out animation you must instead use a **Continue** command. Add a Continue control command that fires when you move to the next story, so the graphic plays its out animation as the next item takes over.

See Continuing a fullscreen graphic below for setup.



Graphics with Transition Logic (TL) built in

The out animation plays on its own when the graphic is taken out. No *Continue* is needed, what matters here is the timing with the next live source. Taking a graphic out clears whatever is on its superchannel at that moment. When you move to the next story, the live source comes up first and the graphic is taken out just after, so if the graphic and the live source are on the same superchannel (SC1), the live source has already taken over that superchannel and the take out also clears the live.



To keep the live on air while a TL graphic animates out, build a preset where SC1 and SC2 are both fullscreen and stacked, with the graphic's superchannel (for example SC2) given visibility priority so it covers the live. Put the graphic on SC2 and the live on SC1. The sequence is then:

1. The graphic is on air fullscreen on SC2, covering the screen.
2. Take the live source fullscreen on SC1. It comes up behind the graphic and stays hidden.
3. Take the graphic out on SC2. The out animation plays and reveals the live on SC1, which stays on air.

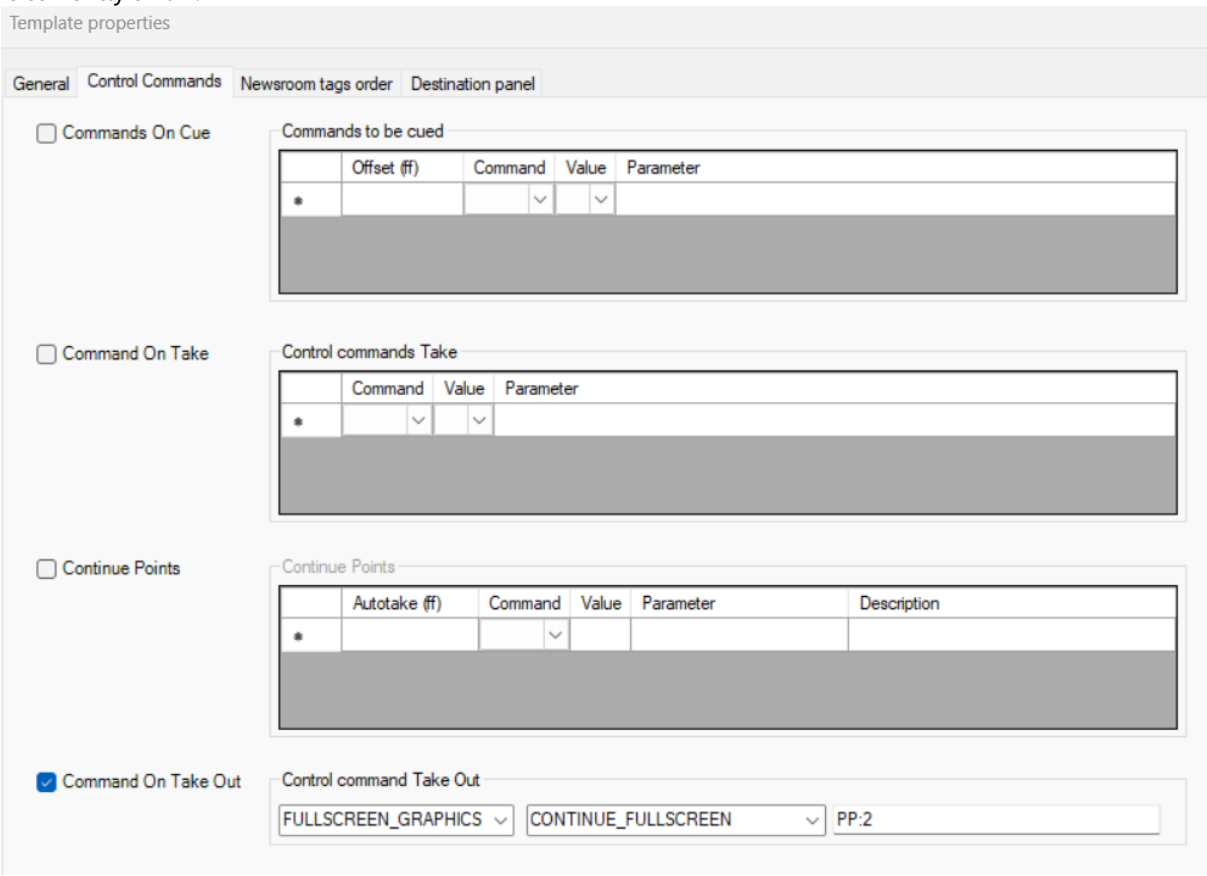
Continuing a fullscreen graphic

Some graphics are built with continue points (stop points) in the scene. The graphic pauses at a stop point until a **Continue** command advances it to the next one. Use this to step through a graphic, for example to reveal the next item or move to the next state, and also to play an out animation at the end. For these graphics you advance with a **Continue** command rather than a take out.

A Continue is sent to the graphic currently on air, on a channel resolved from a mixer effect and a superchannel, and the Media Sequencer advances that graphic to its next stop point.

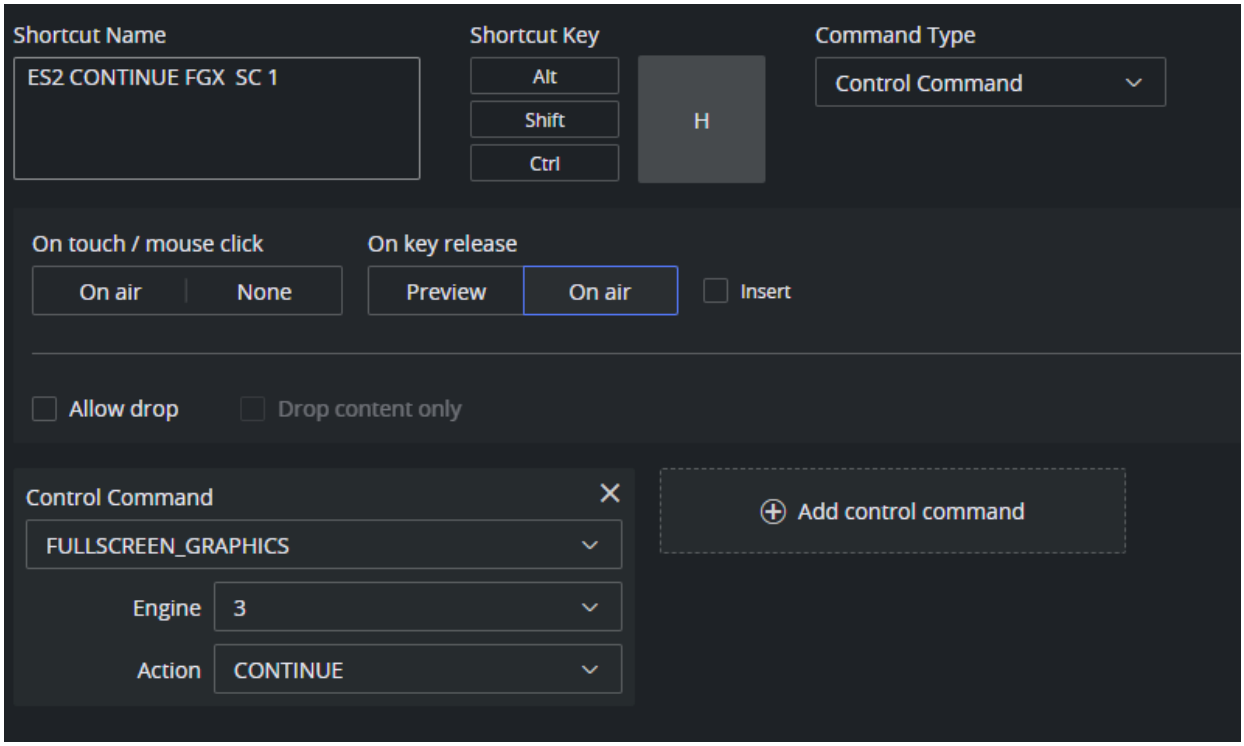
You can trigger Continue two ways:

1. **Control command on a template**
   Add a Control Command, select FULLSCREEN_GRAPHICS, and set the value to CONTINUE_FULLSCREEN. Target a specific mixer effect and superchannel using the format <mixer effect>:<SCn>, or leave it on Current to use the on-air graphic. This is the reliable option, because the target does not depend on which template is currently on air.



2. **Keyboard shortcut**
   In the keyboard editor the field **Engine** lists graphics engine numbers (1 to 8) and **Current**; for ALP only *Current* is valid, since it uses the on-air template's mixer effect.

> **Info**: Currently the editor does not display mixer-effect and superchannel selection, so the keyboard shortcut is limited to the *current* on-air graphic.



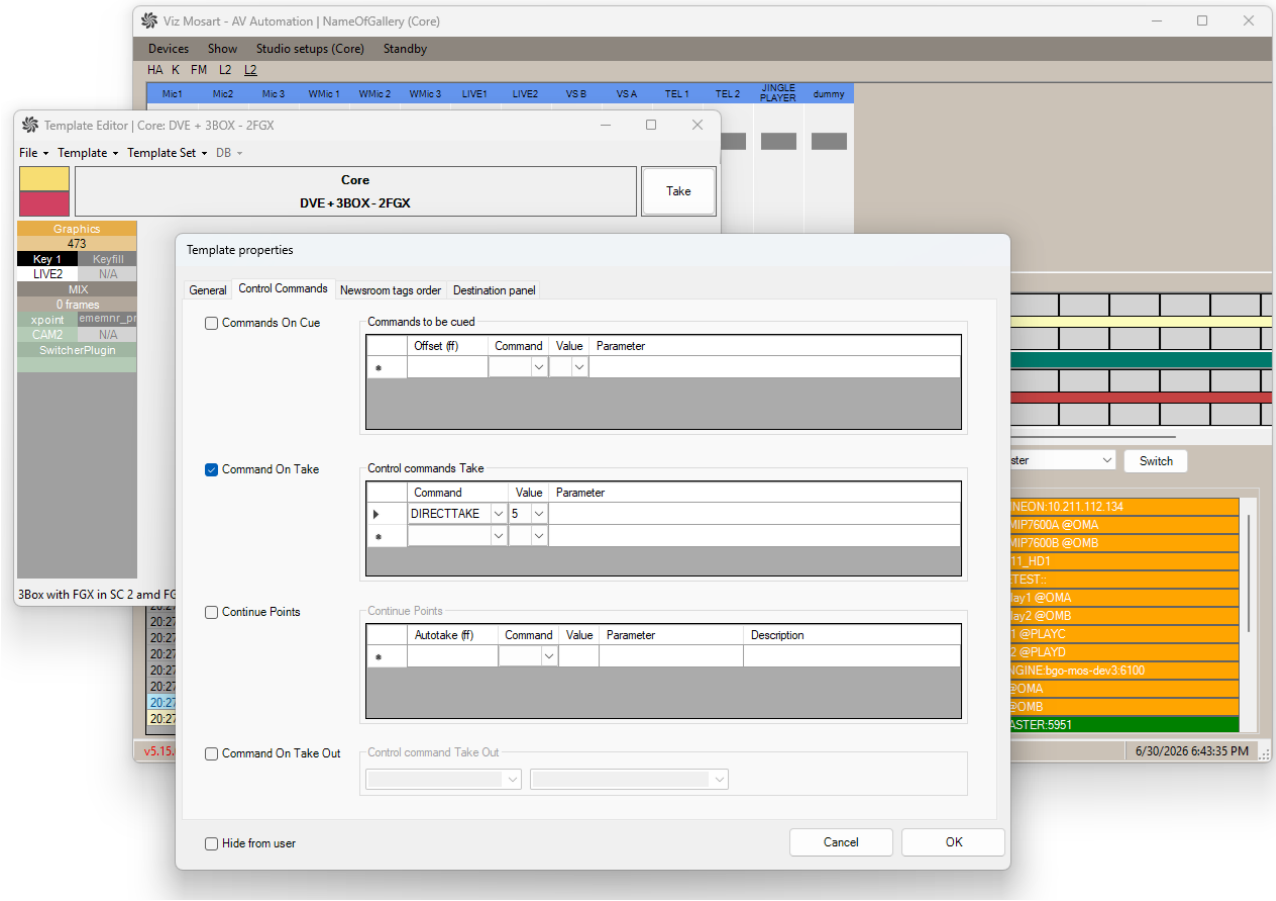
Info: Continue with *Current* only resolves while the graphic is still the on-air graphics item. Once a live-only template (with no graphics) becomes the on-air template, *Current* cannot find the mixer effect and the continue is skipped. Target the mixer effect and superchannel *explicitly on the template control command* to avoid this.

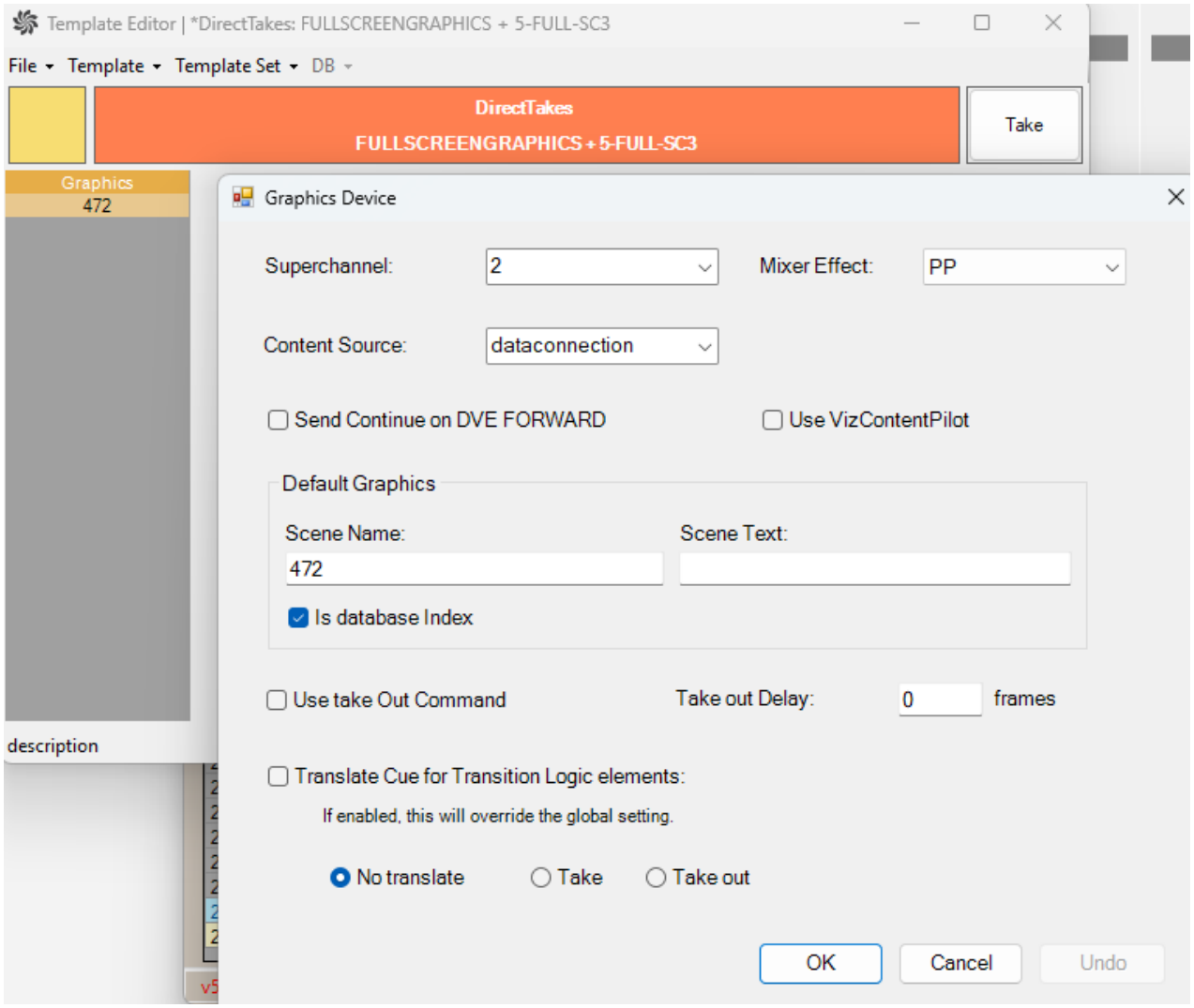
Multiple graphics on DVE box layout

A template can include only one *Graphics* device, so it takes a single fullscreen graphic. To show more than one graphic on a DVE layout, take the extra graphics separately. There are two ways to do this, and in both the graphics are *not taken in sync* (each graphic takes on its own, not in the same frame as the layout or the other graphic).

DirectTake from a control command

On the DVE template, add a control command that direct-takes another template carrying the second graphic. In the Template Editor, on the **Control Commands** tab, add a *DIRECTTAKE* command on take and set its value to the template that takes the second graphic.
The referenced DirectTake template holds the second Graphics device, and its graphic ID must be set in the template itself (Scene Name with Is database Index), so it cannot be driven from the NRCS.


Viz Mosart - AV Automation | NameOfGallery (Core)
Devices
Show
Studio setups (Core)
Standby
Template Editor | Core: DVE + 3BOX - 2FGX
File
Template
Template Set
DB
Core
DVE + 3BOX - 2FGX
Take
Template properties
General
Control Commands
Newsroom tags order
Destination panel
Commands On Cue
Commands to be cued
Offset (ff)
Command
Value
Parameter
Command On Take
Control commands Take
DIRECTTAKE
5
Continue Points
Autotake (ff)
Description
Command On Take Out
Control command Take Out
Hide from user
Cancel
OK
Switch
6/30/2026 6:43:35 PM



Accessory template (secondary item)

Add an *Accessory* template, with its own Graphics device, to the story item that takes the first graphic. In the NRCS, place the accessory as a secondary at the start of the story (or later in the story if needed) so it takes within the DVE item. With this method both graphics, the one on the main item and the one on the accessory, can take their content from the NRCS.

With either method the extra graphic is not taken in the same frame as the layout or the other graphic. Each graphic takes on its own director, so they are not synchronized.

> **Info**: With either method the extra graphic is not taken in the same frame as the layout or the other graphic. Each graphic takes on its own director, so they are not synchronized.

Graphics concepts (via macros)

A graphics *concept* is a design variant of a scene, for example a branded look per show, channel, or region. The same graphics template can render in different concepts without redesigning it.

ALP does not switch concepts on its own: the switcher path takes a graphic in or out but carries no concept. To change the concept on the engines, set the `alternative_concept` value on the relevant Media Sequencer handlers with a user macro, then recall that macro through the standard Vizrt graphics device.

Define the macro in VizrtUserMacros.xml

`VizrtUserMacros.xml` lives in the Vizrt graphics configuration folder and is loaded by the Vizrt graphics device. A macro is a named block of actions; use `<set>` actions to write the concept value on each superchannel (and preset) handler of the engine:

```
<macro name="SPORTS">
  <env>
    <set>
      <path>/config/profiles/ES2Profile/execution_groups/CH1/entry/
alternative_concept</path>
      <value>Sports</value>
    </set>
    <set>
      <path>/config/profiles/ES2Profile/execution_groups/CH2/entry/
alternative_concept</path>
      <value>Sports</value>
    </set>
    <!-- repeat for each superchannel handler (CH3, CH4, ...) of this engine -->
  </env>
</macro>
```

Find the exact paths in the MSE VDOM browser (`http://<mse-server>:8580/app/vdomconfig/vdomconfig.html`), under `/config/profiles/<profile>/execution_groups/<handler>` . Each handler's entry holds the `alternative_concept` value.

Recall the macro from a template or keyboard shortcut

On the template, open **Control Commands > Command On Take** and add:

Command	Value	Parameter
OVERLAY_GRAPHICS	MACRO	SPORTS

The same command can be bound to a keyboard shortcut, so an operator can change the concept live with one key. `FULLSCREEN_GRAPHICS` with `Value=MACRO` behaves the same way. Both are handled by the Vizrt graphics device, which runs the macro against the Media Sequencer.

The macro sets the concept used for graphics taken **after** it runs. To re-skin a graphic that is already on air, recall the macro and then recue (re-take) that graphic.

Target a specific Mixer Effect (PP, ME1, and so on). Each mixer effect is a separate Viz Engine with its own set of superchannels and its own execution groups in the Media Sequencer. A macro targets an engine through the profile and execution-group paths it writes to. To change the concept on ME1 only, point the `<set>` paths at ME1's execution groups; for PP, point them at PP's. A single macro can set several engines at once by listing all their paths.

ALP and macros. The ALP switcher driver does not run user macros, and its superchannel graphics path supports only take-in and take-out, not the MACRO action or concepts.

- Recall concept macros through the standard Vizrt graphics device. Either
  - from a control command or keyboard shortcut bound to that device, or
  - directly from the **Overlay Graphics Interface** application (select the engine and recall the macro).

Because the macro updates the shared Media Sequencer data, the ALP engines pick up the new concept on the next take.

Transitions

ALP supports two type of transitions: *preset* transitions (changing the layout) and *content* transitions (changing what plays on a superchannel). They are independent and can be combined on the same take.

Info: All engines must *share the same content transition names*. Content transition names in A/V Setup (effects) and in the switcher configurations must exist as directors in the preset scenes on every Viz Engine in the setup. A mismatch makes the transition fail silently on that engine.

Preset transitions

A preset transition animates the switch between layouts (for example *1Box* to *2Box*). Viz Engine runs the animation using the transition director defined in the preset scene. The *Switcher Plugin* on the template controls the transition, the *Switcher Crosspoint* transition has no effect on the preset.

Switcher Plugin setting	Preset behavior
No Switcher Plugin	Scene default transition.
Animate unchecked	Instant preset cut.
Animate checked, no duration	Animated transition using the duration last set on that preset transition.
Animate checked + Duration > 0	Animated preset transition with custom duration.

Content transitions

A content transition animates the switch of the live source or graphic on a superchannel. The director must be configured in the Viz Engine preset scene (for example *crossfade*, *bookflip*, *dissolve*).

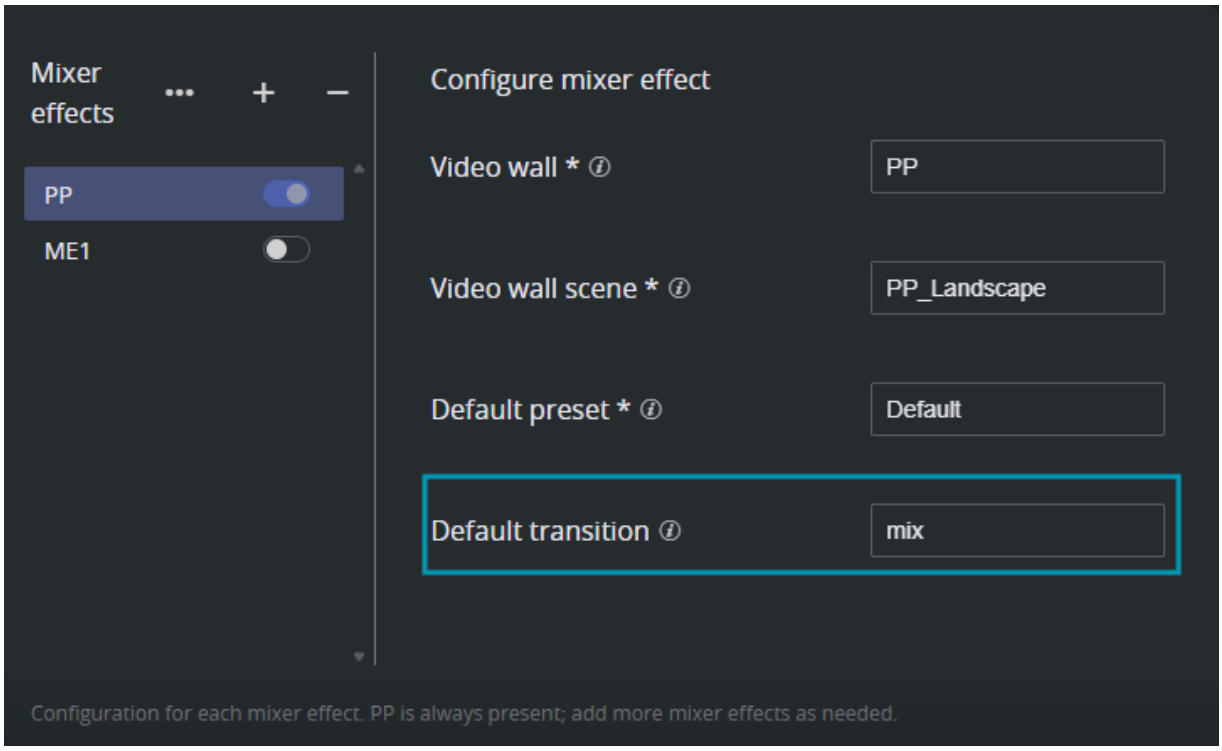
- The main transition control on a template, defined when a *Switcher Crosspoint* is added, controls the transition for superchannel 1 (background).
- Choose main transition control when adding transition on a superchannel where graphics content is taken using the *Graphics* device.
- The *KeyFill* command controls the transition for any superchannel.

Copyright © 2026 Vizrt

Template setting	Content behavior
Cut (or Mix 0)	Hard cut (`___none___`), no content animation.
Mix or Wipe rate > 0	Uses the *Default transition* setting on the switcher configurations.
Effect (from A/V Setup)	Uses the effect name directly as the content transition director name.

Mix or Wipe with rate > 0

For ALP, a *Mix* and a *Wipe* with *rate > 0* are similarly treated as content transition, and behave identically on air. This is configurable in the switcher settings and applies for any transition control on a Mosart template.



To get a distinct wipe, set it up as an Effect in A/V Setup that points at a wipe transition director in the scene.

Effect transitions: when a template uses an effect from A/V Setup, the effect name is used directly as the content transition director name, and the duration comes from the effect's frame count (default *1 second*). Each effect name in A/V Setup must match a director in the Viz Engine scenes.

Designing templates with transitions

Camera template with hard cut (Mix 0): set the template Mix Rate to 0. Mosart sends a hard cut for both preset and content. No animation.

Camera template with transition (Mix > 0): set the Mix Rate above 0 (for example Mix 25 for one second at 25 fps). Mosart uses the configured mix transition for the content, and animates the preset if the *Switcher Plugin* has Animate enabled.

DVE template with preset animation: add a *Switcher Plugin* with the target preset per mixer effect. Enable **Animate** and set the duration. On take, Mosart sends the preset transition as "transition" instead of "cut". Preset animations and content transitions run at the same time, so design the scene to handle both together. Each KeyFill can also carry its own content transition (a Mix rate or a named Effect), set per row in the KeyFills dialog, so box sources can animate independently of the layout.

Template with effect transition: in A/V Setup, create effects whose names match content transition directors in the Viz Engine scenes (for example "bookflip", "dissolve"). On the template, select the effect. Mosart sends the effect name as the content transition director with the configured frame count as duration.

Combining preset and content transitions

Preset and content transitions are independent. The *Switcher Crosspoint* controls what happens to the SC1 source content, and the Switcher Plugin controls how the layout geometry changes. The MSE applies both at the same time. If the preset transition lasts 2 seconds and the content transition lasts 3 seconds, both start together: the layout settles at 2 seconds and the content crossfade completes at 3.

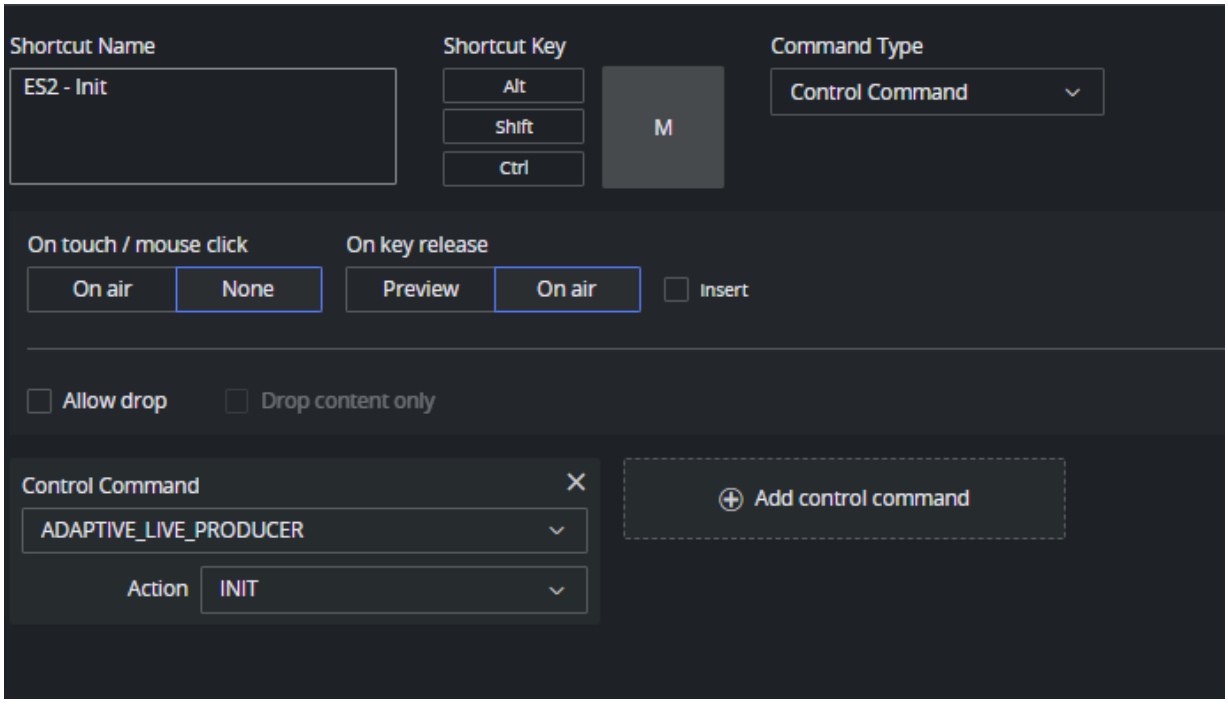
Keyboard Shortcuts

Configure keyboard shortcuts for ALP control commands in Mosart's Keyboard Shortcuts editor. Set the **Command Type** to *Control Command* and select *ENGINE_SWITCHER_VIA_MSE* as the command.

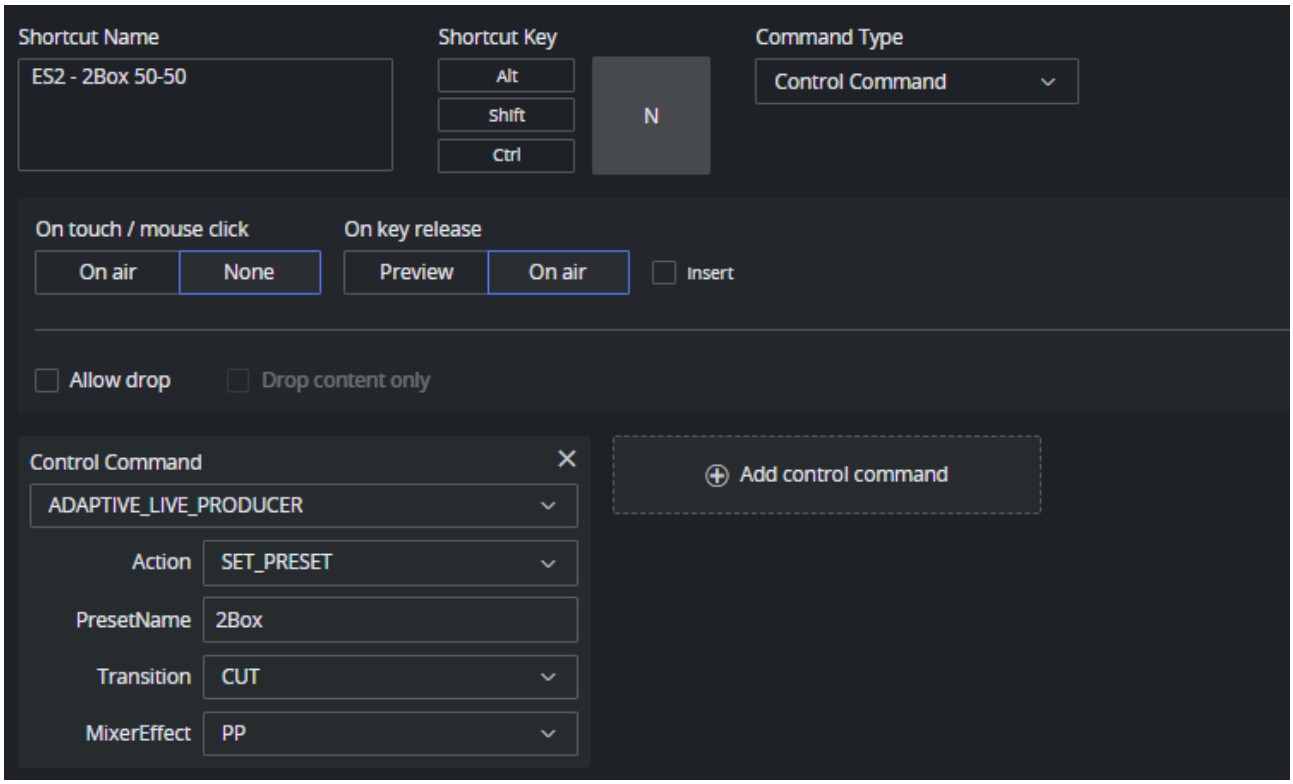
Action	Description
`INIT`	Initialize the ALP connection and show.
`CLEANUP`	Clean up ALP playout state (profile-wide).
`SET_PRESET`	Change preset on a mixer effect (set PresetName and Transition).

Run the **INIT** keyboard shortcut to preload the graphics and scenes used in the show onto the MSE playlist *before the show starts*. This is the main reason to use *INIT*: loading a scene while the show is running can cause a stutter on whatever is currently on air, so every scene that could be played, including heavy ones, should be loaded up-front. *INIT* loads them into the Viz Engine ahead of time, so taking a graphic on air is smooth. After a scene is loaded, its images and texts can be updated in the background from NRCS values without that cost.

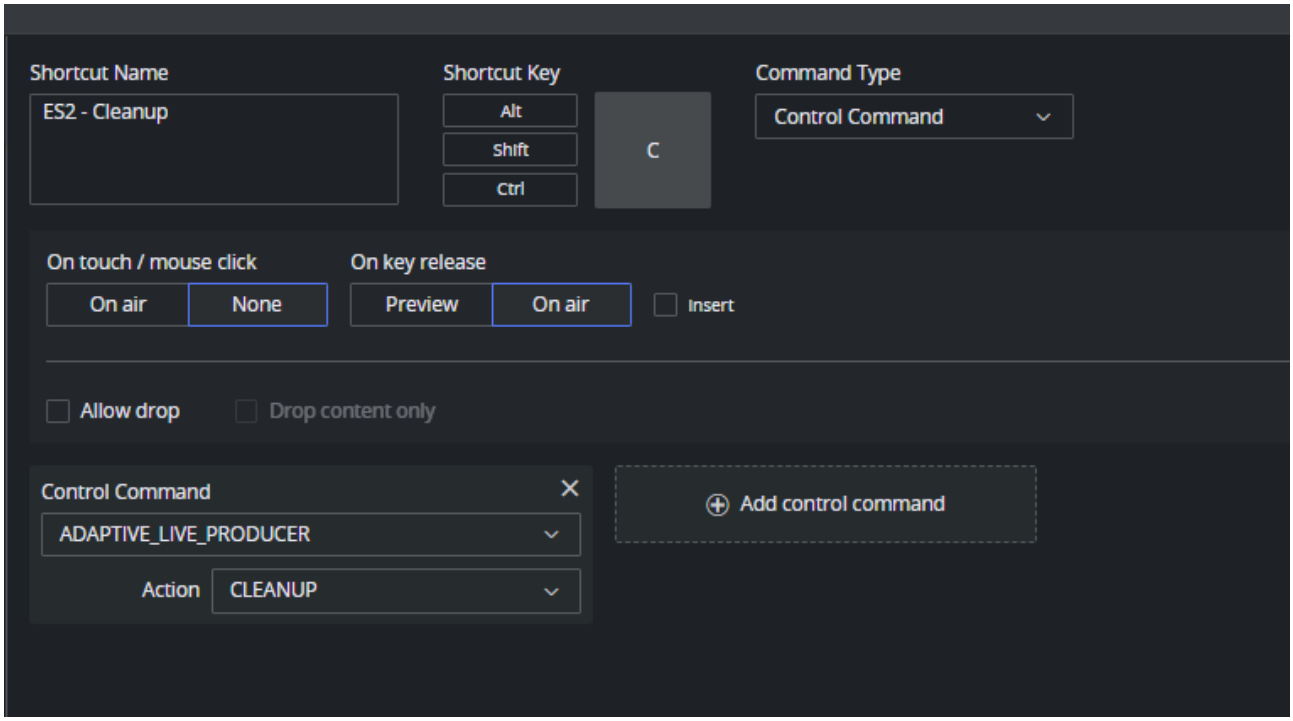
Copyright © 2026 Vizrt



For *SET_PRESET*, set the **MixerEffect** field to target a specific mixer effect (*PP*, *ME1*, *ME2*) or *ALL* to apply to all mixer effects. Set **PresetName** to the target preset and **Transition** to *CUT* or *TAKE* (animated). Taking a preset with a custom animation duration is *not* supported from keyboard shortcuts.



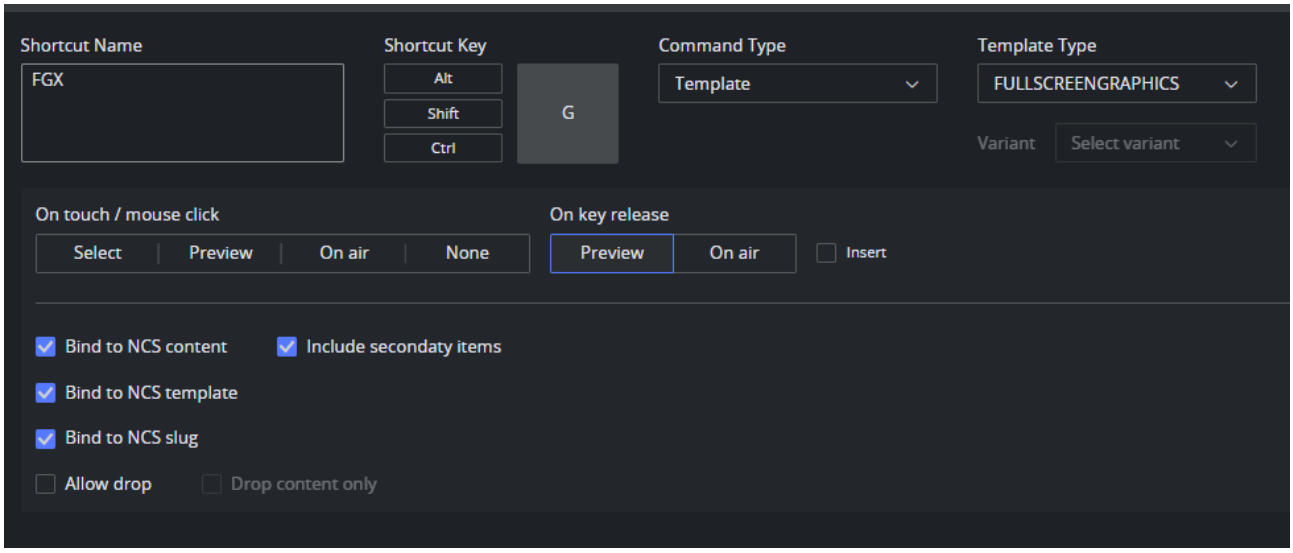
The *CLEANUP* action resets the ALP playout state on the MSE. It operates on the *entire profile* (all mixer effects and all superchannels).
Per-mixer-effect cleanup (targeting only *PP* or only *ME1*) is *not* supported.



CLEANUP is also available as a control command on templates (see below).

> **Note**: *CLEANUP does not clear superchannel content entries*.
> It clears MSE playout state and unloads the preset scene, but superchannel content entries stay in the MSE VDOM, so
> *old content reappears when the scene reloads*.

Beyond the ALP control commands below, the standard Template Keys work as usual for ALP. Keyboard shortcuts that bind a template's content to the NRCS (**Bind to NCS content**) or that drop content onto a key (**Allow drop** and **Drop content only**) behave the same way as for a traditional switcher.



Control commands on templates

ALP actions can be triggered from template control commands (on *cue* or on *take out*).

- In the Template Editor, add a control command.
- Select *ENGINE_SWITCHER_VIA_MSE* as the command type.

Value	Description
`INIT`	Initialize the ALP connection and show.
`CLEANUP`	Clean up ALP playout state (profile-wide).
`SET_PRESET`	Change preset (set the preset name in the Parameter field).

Verify the Full Chain

Once the connection is up (see Verify the Connection) and you have templates, run these checks:

1. **Camera to program**: A Camera template cues in preview and takes to program with the default preset (1Box).
2. **Transition**: A Camera template with Mix 25 crossfades on take.
3. **DVE**: A DVE template with the 2Box preset and keyfills switches several sources together.
4. **Fullscreen graphics**: A fullscreen graphics template shows graphics on the superchannels.

Copyright © 2026 Vizrt

Clean and Dirty Output

- Clean and Dirty workflows in ALP
- Engine Setup
- Overlay Graphics

This page describes how to produce a *clean feed* alongside the *dirty (program) feed* with Adaptive Live Producer (ALP).

Clean feed is the program picture without overlay graphics. It can already contain composited live sources, presets, DVE boxes and fullscreen graphics; a clean feed is not the same as *no graphics*. *Clean* is the absence of the overlay graphics that produce the dirty feed. Clean feeds are typically used for distribution, archiving, and downstream regionalization where overlays are later added.

Dirty (program) feed is the clean picture with overlay graphics (branding, name straps, lower thirds) added on top. It is the main program feed sent to air.

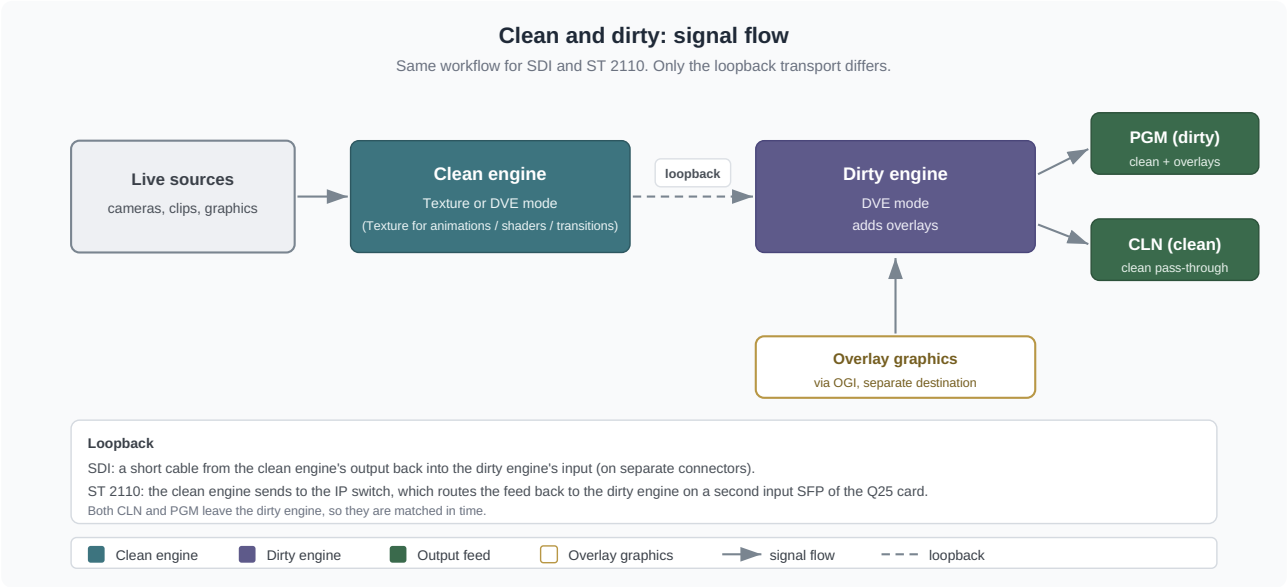
Clean and Dirty workflows in ALP

Clean and dirty output is produced with *two Viz Engine* instances:

- The *clean engine* renders the clean program (live sources, presets, DVE boxes, fullscreen graphics and transitions).
- Its output is *looped back* into the *dirty* engine.
- The *dirty engine* adds the overlay graphics and outputs both the clean (CLN) and dirty (PGM) feeds.

> **Info**: The two engine instances can run on the *same machine* or on *two separate engine machines*. Both options work the same way.

Because both the CLN and PGM feeds are produced by the dirty engine, they stay *matched in time*. A single engine cannot produce a clean feed (with graphics) and a dirty feed at the same time, which is why two engines are used.



Engine Setup

The same workflow is used for SDI and ST 2110:

- **Clean engine.** Renders the clean program and sends its output to the loopback. Runs in Texture mode when the clean program needs animations, shaders or custom transitions; runs in DVE mode for more frame stability when those are not required. DVE is more guaranteed to deliver every frame, Texture is less certain if the Matrox board is heavily loaded. DVE has lower in-to-out delay (3 frames).
- **Dirty engine.** Runs in *DVE mode*. Takes the looped-back clean feed on one input, adds the overlay graphics, and outputs the CLN and PGM feeds **in sync** from its Matrox clean and program outputs.

SDI

- Use an *X.mio* board. The board model depends on the number of sources and performance required.
- Loop the clean engine's output back into the dirty engine's input over a cable. The clean engine output and the dirty engine input must be on *separate connectors*.

ST 2110

- Use a Matrox **X.mio5 Q25** board (two SFP pairs). An IP switch will not return a signal on the SFP pair it was sent from, so the clean feed must be received on the *second SFP pair*. On the card, connectors *1* to *16* are on the first SFP pair and *17* to *32* on the second. Keep the clean engine output and the dirty engine input in *separate ranges* (one in 1 to 16, the other in 17 to 32).
  This setup needs the *Q25* variant for its two SFP pairs. Use the **X.mio5 Q25** when the dirty engine runs in DVE mode (the standard setup). The **DSX.le5 Q25** does not support DVE, so use it only when the dirty engine runs in Texture mode.
- Both engines can subscribe to the *same multicast* for shared sources, so a shared source costs no extra bandwidth.

NDI

ALP supports NDI, but not in this clean and dirty loopback configuration. Use SDI or ST 2110 for clean and dirty output.

> **Info:** Supported video format
> HD (1080i / 1080p). This setup is not sized for UHD. At UHD, each video signal is about four times the data of HD, and the Matrox card's I/O bandwidth and the engine's compositor performance are likely to be exceeded by the dual-engine and clean-feed output combination.
> Each engine uses its own input connectors (inputs are not shared between the two engines).

Overlay Graphics

Overlay graphics are handled the standard way, through the Overlay Graphics Interface (OGI) in Viz Mosart, and are taken on the *dirty engine* on a separate destination, not on the superchannels.

> **Info**: An exception is an *overlay that must be part of the clean feed*.

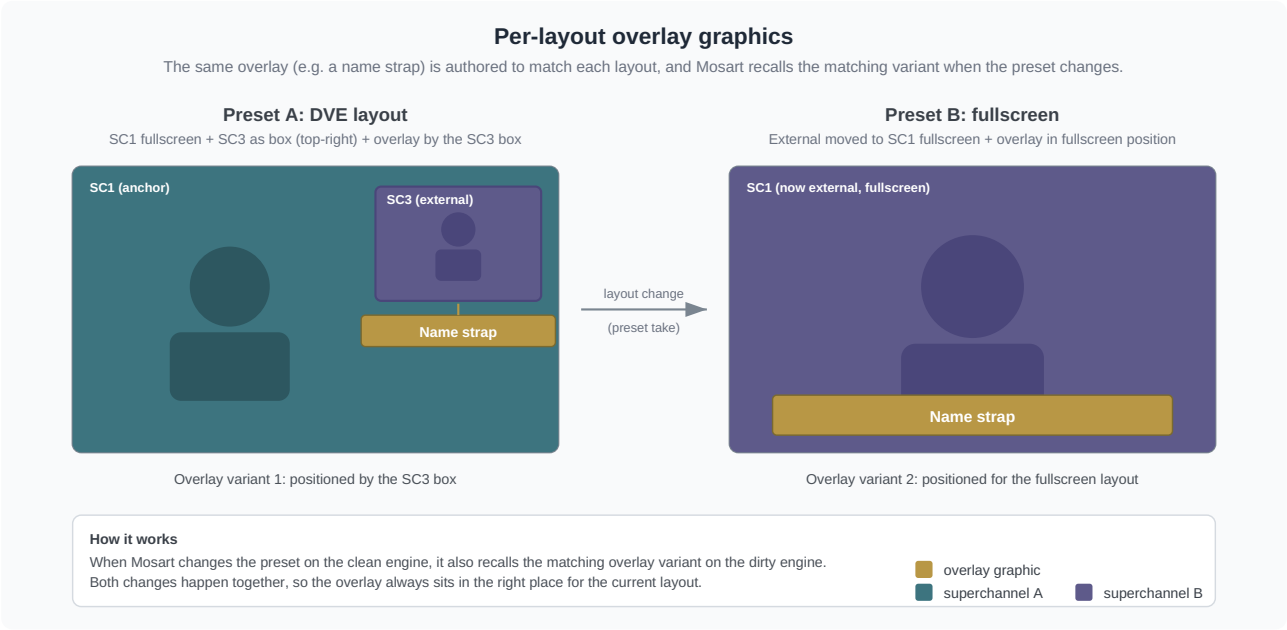
> Always place this one on a superchannel in the mixer on the clean engine (a superchannel reserved for overlays), so it is composited into the clean output.

Author one overlay graphic scene (or a variant inside a scene) for each layout and position. An overlay can sit by a box in one layout and move to the fullscreen position in the next; a matching overlay scene is created for each case. Mosart recalls the matching overlay whenever the layout (preset) changes on the clean engine, so the overlay and the layout change together.

Example

1. A DVE preset has SC1 fullscreen and SC2 and SC3 as boxes. To label the source in the SC3 box, take an overlay designed for the SC3 box position.
2. Take a fullscreen preset where the SC3 source moves to fullscreen on SC1. Recall a different overlay, designed for the fullscreen position.

Both overlays are taken through OGI as usual; Mosart recalls the matching overlay when the layout changes.



Tally in the Adaptive Live Producer

- Setup
- Tally Rules by Template Type
- ALP-specific Limitations

Viz Engine is not a traditional switcher, so it produces no tally of its own. Viz Mosart fills the gap. Because Mosart commands every crosspoint, it can track which sources are on program (red) and cued in preview (green), sending details to multiviewers over TSL UMD v3.1.

Setup

Tally output (TSL UMD v3.1) is a standalone Viz Mosart feature. This setup covers behavior specific to Adaptive Live Producer.

> **Info**: For general details on how to enable Tally, the host/port arguments, the packet format, and the TSL UMD simulator, see section Tally.

Tally Rules by Template Type

Crosspoint numbers from A/V Setup map directly to TSL UMD addresses on the multiviewer. What gets set or cleared depends on the template:

Template type	Clears previous tally?	Sets new tally?
Has Switcher Crosspoint (Camera, Package, Live, DVE)	Yes. Sources not in the new template are turned off.	Yes
KeyFills only, no crosspoint (DVE overlay)	No. Background is unchanged.	Yes (keyfills)
No crosspoints (Jingle, Break)	No	No, persists
Fullscreen Graphics (no crosspoint, has graphics)	Yes. Graphics replace the full output.	No new video sources
M/E templates (SetMeInPP)	Yes. Switching away from an M/E clears its sources.	Yes
Preset change only (SET_PRESET, no crosspoint)	Yes. Sources on superchannels the new preset hides are turned off.	No new sources. Sources on superchannels the preset keeps visible stay red. If the scene has no layout information for the preset, tally is left unchanged.

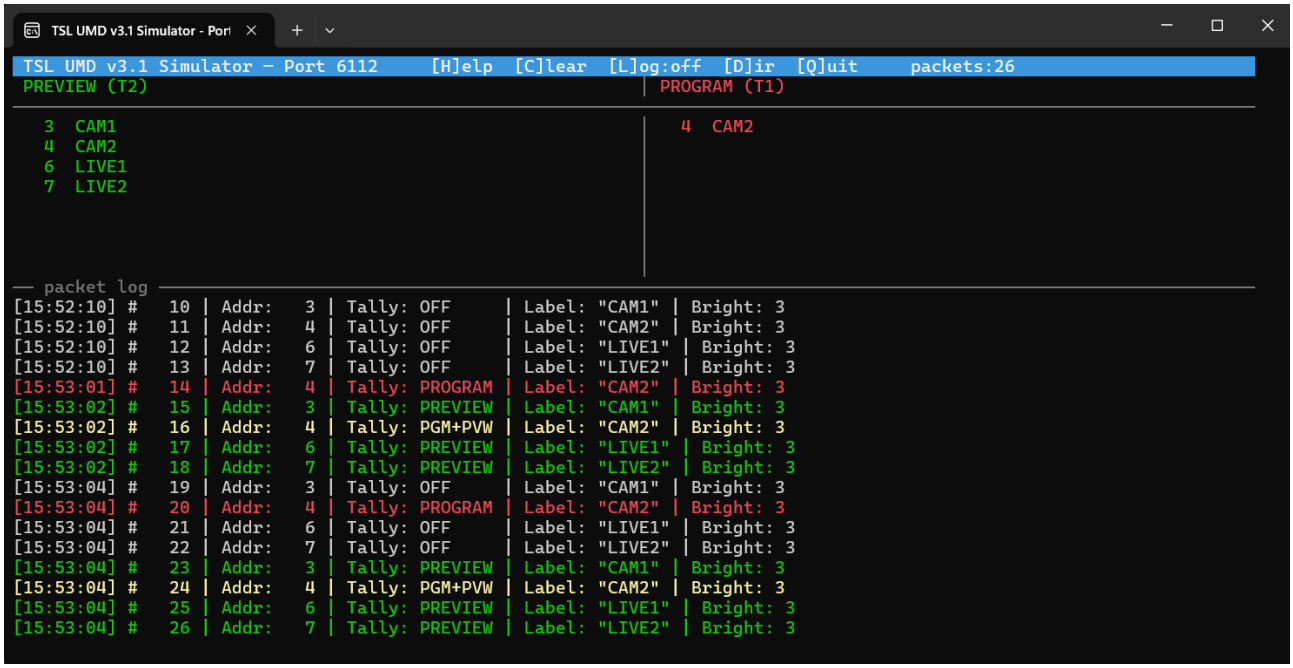
Copyright © 2026 Vizrt

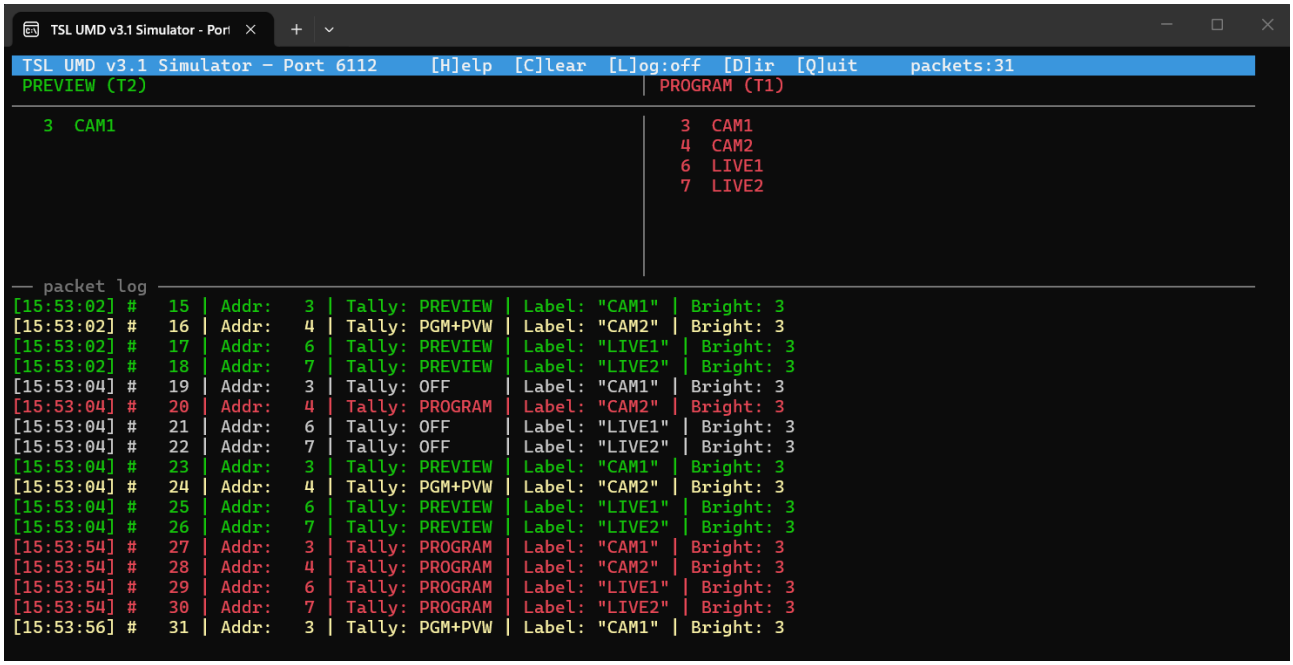
Template type	Clears previous tally?	Sets new tally?
Preview (Cue)	Previous preview cleared	Yes (green)
Direct changes / Control Commands	Only the replaced source	Yes (immediate)

Examples

Camera to DVE 3Box to Camera3

Step	Action	Tally
1	CAM1 on air	CAM1 = Red
2	Take DVE 3Box (CAM1 + LIVE1 + LIVE2)	CAM1, LIVE1, LIVE2 = Red
3	Take Camera3	CAM3 = Red. CAM1, LIVE1, LIVE2 = Off





Changing preset (keyboard shortcut)

Changing only the preset reshapes the layout. Sources on boxes the new preset drops turn off; sources on boxes it keeps stay red, and no new source is added. Here, 3Box (CAM1 + LIVE1 + LIVE2) changes to 2Box, which keeps the first two boxes:

Step	Action	Tally
1	DVE 3Box on air (CAM1 + LIVE1 + LIVE2)	CAM1, LIVE1, LIVE2 = Red
2	Change preset to 2Box (SET_PRESET keyboard shortcut, no crosspoint)	CAM1, LIVE1 = Red. LIVE2 = Off

Fullscreen Graphics

Step	Action	Tally
1	LIVE3 on air	LIVE3 = Red
2	Take fullscreen graphics (no crosspoint)	LIVE3 = Off. No new tally

ALP-specific Limitations

Tally is based on the Program/Preset (PP) M/E only. Sources on other M/Es are not tracked independently. Only T1 (program) and T2 (preview) are sent.

Copyright © 2026 Vizrt

Troubleshooting

- Files to Collect
- Context to Include

When reporting an Adaptive Live Producer (ALP) issue, collect the files below from the Mosart Server machine and the MSE host, then add the context notes that follow.

Files to Collect

File	Location	How to collect
Instrumentation log	`Custom`	In AV Automation, open **Show > Instrumentation Panel**. Run the rundown to reproduce. Save the CSV from there.
Application log	`C:\MMLogs\`	XML log with full event details. Collect the log from the time of the issue.
MSE log	MSE host	MSE internal log showing superchannel state, arm/fire commands, and engine communication.
Rundown XML	`C:\manus\`	The NRCS rundown loaded when the issue occurred.
channeltemplates.xml	`C:\channeltemplates\`	Template definitions including crosspoints, keyfills, and presets.
avconfig.xml	`C:\channeltemplates\`	Video source mapping, engine presets list, and effects configuration.
ALP settings	Settings app (export) / `C:\ProgramData\Mosart Medialab\ConfigurationFiles\`	The ALP configuration from the **Settings** app.
Tally log (if needed)	`C:\MMLogs\`	Run TslUmdSimulator with logging (the 'L' command) to capture tally output during the issue.

Copyright © 2026 Vizrt

Context to Include

1. What story or template was on air when the issue happened.
2. What action triggered it (take, cue, direct take, keyboard shortcut).
3. Expected result versus actual result.
4. A screenshot or video if the issue is visual.
5. Whether the issue is reproducible.

Copyright © 2026 Vizrt

12.2 Media Sequencer

This section introduces the most recent developments for graphics control managed by Vizrt's middleware component, Media Sequencer (MSE). It also introduces main/backup switching of the Media Sequencer.

- Configuring Vizrt Graphics
- Main/Backup Topology

12.2.1 Configuring Vizrt Graphics

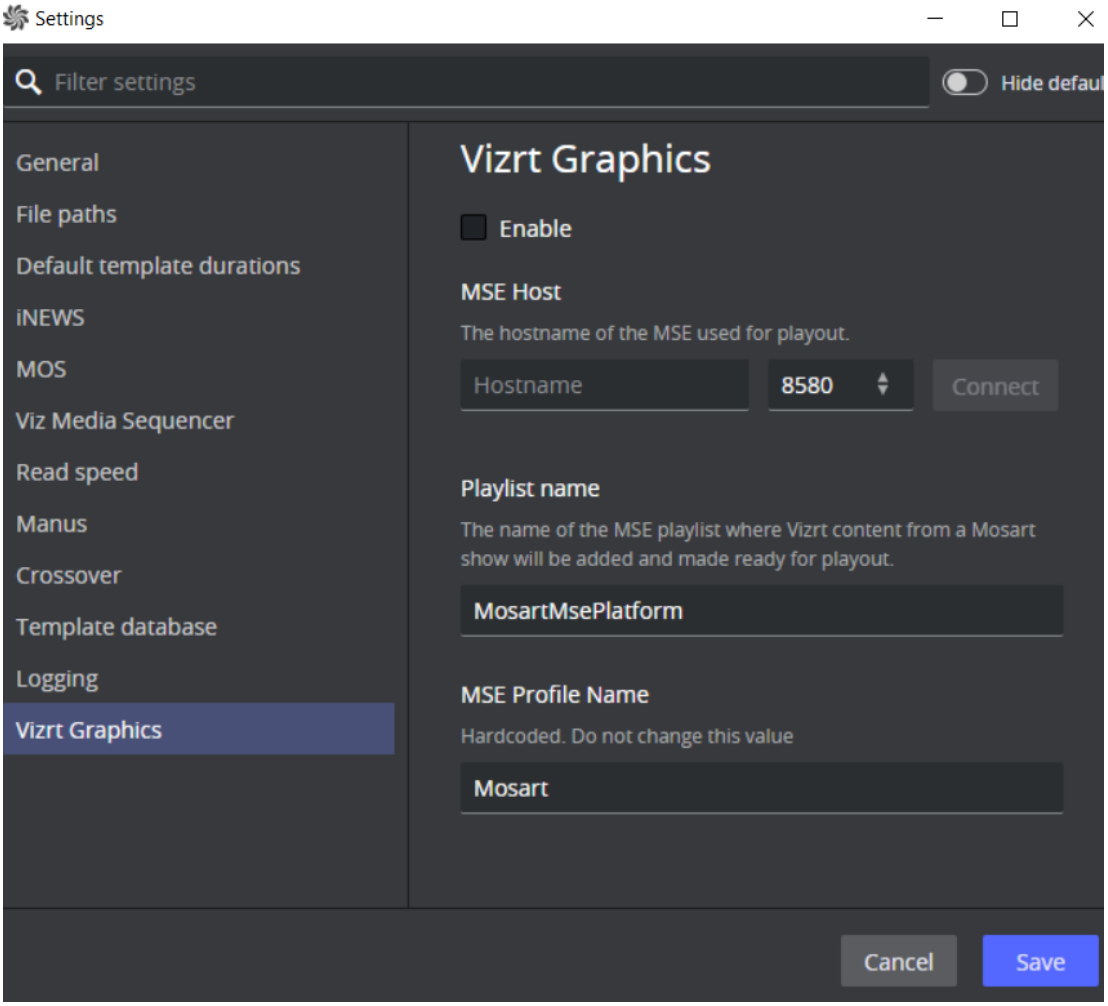
A Viz Mosart driver communicates with the MSE. MSE-driven functionality is being continually improved to control various graphic elements.

- From the AV Automation menu bar, navigate to **Devices > Properties > Vizrt Graphics tab** and follow the **Vizrt Graphics** link.

The current supported graphics are listed below:

Viz Arc Elements

- Use the **Settings** menu to set up Vizrt Graphics.



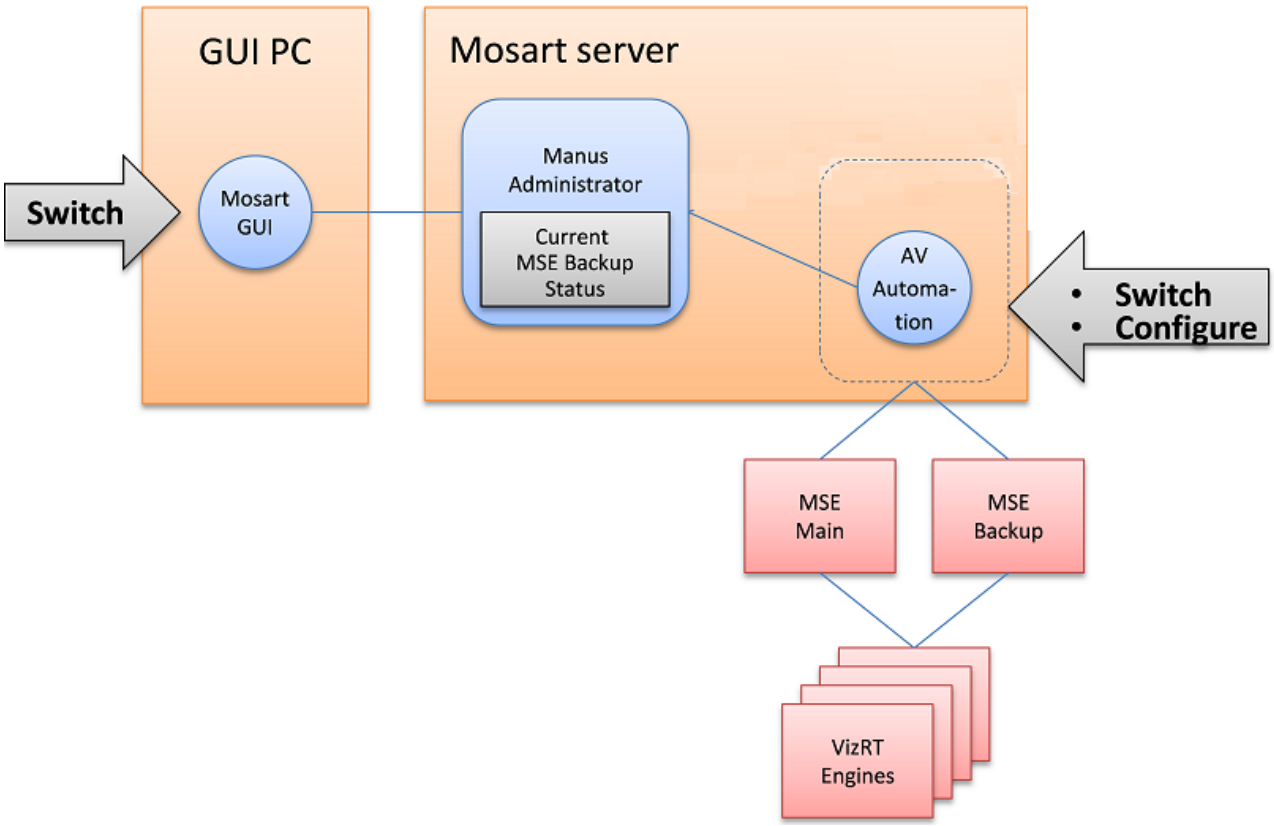
12.2.2 Main/Backup Topology

Selecting the Media Sequencer

- It is possible to select the *main* or *backup* Media Sequencer, from Viz Mosart. See Media Sequencer#Switching Media Sequencer from the Viz Mosart GUI.
- There may be a delay of several seconds after switching from one to the other, before the Media Sequencer responds to Viz Mosart commands.

Configuration Alternatives

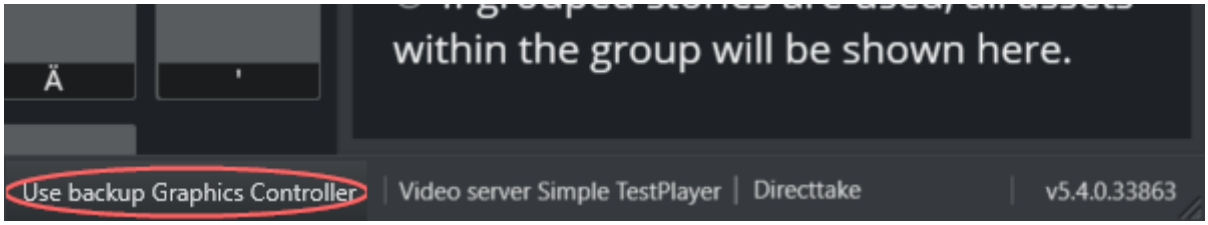
The diagrams below show the configuration and operations possibilities of the various components in the Media Sequencer backup concept.
Gray arrows show which operations are available in each Viz Mosart module.



To configure MSE backup

- See section Redundancy.

Switching Media Sequencer from the Viz Mosart GUI



Switching between main and backup Media Sequencer is normally done in the Viz Mosart GUI.

To Switch Media Sequencer

- At the bottom right corner of the Viz Mosart UI is the text: **Use backup Graphic Controller**.
  This text is clickable and switches Viz Mosart to the backup Media Sequencer.

Once switched, a warning text on yellow background, **Use main Graphics Controller** visibly alerts to the exceptional state.

During the switching process, the alarm indicators in AV Automation momentarily go red, but return to green when the switching process is finished.

Note: The Media Sequencer requires several seconds to initialize all parts, so it is recommended to wait some time before continuing graphics operations.
The exact delay varies with graphics size and complexity.

12.3 Mosart Remote Control REST API

- Getting started
  - HTTPS
  - Authentication
  - Main/Backup setup
  - Swagger Documentation
- Endpoints
  - Assets
  - Devices
  - Media
  - Settings
  - Status
  - Timeline

The Mosart Remote Control Service (RCS) provides a REST API for controlling Viz Mosart rundowns using REST.

12.3.1 Getting started

To begin using the Mosart REST API

1. Copy `RemoteDispatcherServiceConfig.xml` in `C:\Program Files (x86)\Mosart Medialab\Mosart Server\ConfigurationFiles`
   to
   `C:\ProgramData\Mosart Medialab\ConfigurationFiles`.
2. Open the copied file in a text editor.
   a. Locate the `<item>` element with `name="RestApiKey"`
   b. Set **value** an arbitrary string that you choose.
   All API clients must include this key to authorize their requests.
   For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RemoteServiceConfig>
  <Properties>
    ...
    <item name="RestApiKey" value="test-api-key" />
```

   c. Save the file.
3. From Windows Services, restart the **Mosart Remote Control Service**.
4. Make an HTTP request to http://[machine-name]:55167.
   For example:

```
curl -X GET "http://localhost:55167/api/v1/command/reload" -H  "accept: */*" -H
"X-API-Key: test-api-key"
```

HTTPS

If you want to enable API access over HTTPS, you must edit the file `RemoteDispatcherServiceConfig.xml`, changing the following items:

```
<?xml version="1.0" encoding="UTF-8"?>
<RemoteServiceConfig>
  <Properties>
    ...
    <item name="SignalRUseHttps" value="true" />
    <item name="SignalRCertificatePath" value="C:\cert\server.cer" />
    <item name="SignalRCertificateKeyPath" value="C:\cert\server.key" />
    <item name="SignalRCertificatePassword" value="password" />
```

Authentication

Note: When configuring authentication for the Mosart Remote Control REST API, you must also make corresponding authentication configurations for any Mosart Web Apps you are using.
Refer to the topic **Configuration Tool > Authentication** in the Mosart Web Apps User Guide.

1. (Prerequisite) Any REST API authentication first requires a configured HTTPS connection, as described above.
2. Create a new Client/App registration according to your authentication provider guidelines:
   a. Keycloak guide
   b. Keycloak guide for Vizrt SSO
   c. Entra guide
   d. Okta guide.
3. On the Mosart server, modify the file `RemoteDispatcherServiceConfig.xml` for the following items:

```
<item name="UseOidcAuthentication" value="true"/>
<item name="OidcAuthority" value="{{Your Authentication Server}}"/>
<item name="OidcAudience" value="{{Your Audience}}"/>
```

With the authentication enabled, you now always need to send a JWT token (from the Authentication server), either as a header or a URL parameter in your REST calls, for them to authenticate.

Main/Backup setup

If you want your API calls to go to the currently active Mosart Server, you can route them through *Mosart Web Applications*.

Install them on a separate machine and configure connection to both main and backup. Then use the endpoint http://[web-apps-machine-name]:65152 and Web Apps will act as a proxy and redirect the call to the active server.

Copyright © 2026 Vizrt

Swagger Documentation

Mosart Server ships with an interactive Documentation page, available after enabling the Mosart REST API.

Explore interactively: Swagger UI (local instance).

12.3.2 Endpoints

Below is a description of the various Endpoints available through this API. They are grouped according to category:

- Assets
- Devices
- Media
- Settings
- Status
- Timeline.

Assets

List Graphics

GET `/api/v1/assets/graphics?onair={true|false}` *(since v5.13.0)*

Get a list of all the overlay graphics in a rundown.
The optional *onair* parameter filters to only graphics currently on air. *(since v5.14.0)*

Example of the response.


```
[
    {
        "id": "100_a1e1e678-4adc-420a-bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-
bdbe-54b81b3c25a6_2",
        "type": "100",
        "variant": "TIMECODE-DSK",
        "slug": "Mosart=L|-00:00|00:29",
        "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
        "status": 0,
        "graphicType": "",
        "handlerName": "DSK",
        "description": "(DSK) - Mosart=L|-00:00|00:29",
        "hasContent": true,
        "in": 0,
        "duration": 29,
        "plannedDuration": 29,
        "actualDuration": 0,
        "fields": [],
        "hasTemplate": null,
        "emptyTemplate": null,
        "templatePlaceHolders": null
```

```
    },
    {
        "id": "100_a1e1e678-4adc-420a-bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-
bdbe-54b81b3c25a6_5",
        "type": "100",
        "variant": "MANUAL-DSK",
        "slug": "Mosart=L|M|O",
        "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
        "status": 1,
        "graphicType": "MANUAL",
        "handlerName": "DSK",
        "description": "(DSK) - Mosart=L|M|O",
        "hasContent": true,
        "in": 0,
        "duration": 5,
        "plannedDuration": 5,
        "actualDuration": 0,
        "fields": [],
        "hasTemplate": null,
        "emptyTemplate": null,
        "templatePlaceHolders": null
    }
]
```


Take/Take out Graphics

POST `/api/v1/assets/graphics/{mosartItemId}/take` *(since v5.13.0)*

POST `/api/v1/assets/graphics/{mosartItemId}/take-out` *(since v5.13.0)*

Take or take out an overlay graphics by its Mosart ID.

`mosartItemId` : The ID of the overlay graphic you want to take out. `id` property from `/graphics` .

POST `/api/v1/assets/graphics/take?name={name}` *(since v5.13.0)*

POST `/api/v1/assets/graphics/take-out?name={name}` *(since v5.13.0)*

Take or take out an overlay graphics by its name.

`name` : The **slug** of the graphic. This is the name visible in the Viz Mosart GUI. `slug` property from `/graphics` .

> **Info**: If there are multiple graphics with the same name, only the *first* one in the rundown will be picked.

Take Story Item from rundown

POST `/api/v1/assets/template/{mosartItemId}/take?target={program/preview}` `insert={true/false}` *(since v5.14.0)*

Take a story item ("filled" template) from a rundown by its ID into which *target* (Preview or Program). For target *Preview*, an *insert* option governs whether the template shall be inserted in front of, or overwrite, an existing

manual item.
When an optional parameter is not used, defaults are *program* (for target) and *false* for insert.

Devices

Standby Device

PATCH `/api/v1/devices/{type}` *(since v5.13.0)*

Sets the standby status of the specified device type.

`type` : audio, audio-player, fullscreen-graphics, generic-rest, gpi, graphics, lights, loudness, robotic-camera, router, subtitling, switcher, video, video-wall, virtual-set, weather.-graphics, generic-rest, graphics, robotic-camera

Example of the request body

```
{
  "standby": true
}
```

PATCH `/api/v1/devices/{type}/{id}` *(since v5.13.0)*

Sets the standby state for a single device/controller identified by one ID.

`type` : fullscreen-graphics, generic-rest, graphics, robotic-camera

`id` : The device (or controller) ID. For graphics, this can be the engine ID or the destination.

Example of the request body

```
{
  "standby": true
}
```

PATCH `/api/v1/devices/{type}/{controllerId}/{deviceId}` *(since v5.13.0)*

Sets standby for a device addressed by controller and device IDs.

`type` : robotic-camera

`controllerId` : The controller ID.

`deviceId` : The device ID.

Example of the request body

```
{
```

```
  "standby": true
}
```

Media

Search media

GET `/api/v1/media/search?name={clip-name}` *(since v5.11.0)*

Searches the video servers configured in Mosart and returns a list of clips.

`clip-name` : The word you want to search for.

Example of the response.

```
[
  {
    "name": "File name @100%[test]&copy.mxf",
    "description": "File name @100%[test]&copy.mxf",
    "previewPath": "http://vmp-superdemo:21099/serve/proxy/34c8bc7b-c5a2-4c0f-
ad69-50452346cb10.mp4",
    "thumbnailPath": "http://vmp-superdemo:21099/serve/thumb/34c8bc7b-c5a2-4c0f-
ad69-50452346cb10.jpg"
  }
]
```

Settings

Get NRCS settings

GET `/api/v1/settings/nrcs` *(since v5.13.0)*

Get the details of how Mosart is configured to connect to the NRCS.

Example of response.

```
{
  "ncsType": "Mos",
  "mosServer": "ws://mosserver/api/v1/showmaker/mos/mymossocketncs",
  "mosServerBackup": "",
  "ncsId": "SHOWMAKER.MOS",
  "ncsIdBackup": "",
  "mosId": "MOSART.MOS.SHOWMAKER",
  "useBackServer": false,
  "mosCommunicationMode": "WebSocket",
  "forceMosProtocolVersion": "2.8.4",
  "mosLowerPort": 10540,
  "mosLowerPortTimeout": 0,
```

```
  "mosUpperPortTimeout": 0,
  "mosHighPortTimeout": 0,
  "mosServerTimeout": 0,
  "mosServerTimeZone": "",
  "mosSchema": "http://www.mosartmedialab.no/schema/mositem.dtd",
  "defaultReadyToAir": false,
  "templatesToMosObject": [
    "Enabled",
    "SendAllSets"
  ],
  "storyScope": "Single",
  "mosReplyWithConnectionMosId": false,
  "minTimeBetweenStoryUpdate": 1000,
  "itemIdGenerator": "",
  "ignoreInitialSynchronization": false,
  "ignoreSendItemStatusFilter": "",
  "newsRoomTagKeepKeywords": [],
  "itemStatusUnavailableValues": [
    "NOT READY",
    " ERROR"
  ],
  "videoServerGotoDelayOnRetake": 1500,
  "ftpServerName": "inews-server",
  "ftpServerUserName": "inews-user",
  "ftpServerPassword": "inews-password",
  "ftpPort": 21,
  "ftpUseActiveMode": true,
  "ftpActivePortRange": "",
  "ftpPreferEncryption": true,
  "inewsWorkingDirectory": "SHOW",
  "inewsEncoding": "UTF8",
  "inewsWebServiceConnectionString": "",
  "inewsUseFullStoryLocator": false
}
```

Update NRCS settings

PATCH `/api/v1/settings/nrcs` *(since v5.13.0)*

Change the NRCS connection in Mosart. Can we used to switch between different NRCS systems. I.e. using Showmaker for certain shows.

> If the name of the setting (the *key*) does not exist -> that setting will be skipped, while the rest will succeed. If the *value* is of the wrong type -> **400 Bad Request**.

Example of body.

```
{
  "ncsType": "Mos",
  "mosServer": "ws://mosserver/api/v1/showmaker/mos/mymossocketncs",
  "ncsId": "SHOWMAKER.MOS",
```

```
  "mosId": "MOSART.MOS.SHOWMAKER",
}
```

Status

Build

GET `/build` *(since v5.1.0)*

GET `/api/v1/build` *(since v5.4.0)*

Information about the version of the Mosart Server.

Example of response.

```
{
  "version": "5.13.0.38555 Beta",
  "timestamp": "2025-10-21T07:15:10Z",
  "hubVersion": 1
}
```

Status

GET `/status` *(since v5.4.0)*

GET `/api/v1/status` *(since v5.4.0)*

Get the status of the Mosart server.

Example of returned status

```
{
    "state": "Active",
    "timeline": "Running",
    "autoTake": false,
    "rehearsalMode": false,
    "crossoverClient": false,
    "serverDescription": "MosartServer"
}
```

Timeline

Timeline Information

GET `/api/v1/timeline` *(since v5.13.0)*

Get timeline information including **timeline status**, **current story**, **next story**, **current item** and **next item**.

Example of response.

```
{
  "status": "Stopped",
  "currentStory": {
    "id": "1F52BA57",
    "slug": "S1"
  },
  "nextStory": {
    "id": "1E52BA57",
    "slug": "S3"
  },
  "currentItem": {
    "id": "0_SHOW.TESTSHOW.T-1_1F52BA57_CAMERA20_6",
    "slug": "CAMERA 20"
  },
  "nextItem": {
    "id": "0_SHOW.TESTSHOW.T-1_1E52BA57_CAMERA2_16",
    "slug": "CAMERA 2"
  }
}
```

Update source item in preview or program

PATCH `/api/v1/timeline/{target}` *(since v5.14.0)*

Content-Type: application/json
Updates fields on the current item in preview or in program. Used to change the crosspoint assignment for a newsroom tag field.

Example of body

```
{
"newsroomTag": "left",
"crosspoint": "INPUT 1"
}
```

Rundown

GET `/api/v1/rundown` *(since v5.13.0)*

Get the currently loaded rundown.

Example of response.

```
{
    "id": "a1e1e678-4adc-420a-bdbb-8aaa76dc6cf0",
    "name": "graphics",
```

```
    "stories": [
        {
            "id": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
            "insertId": "NSID-1",
            "storyDuration": 10,
            "storyPlannedDuration": 0,
            "storyBackTime": -1,
            "pageNumber": "",
            "slug": "Untitled",
            "accessories": [
                {
                    "id": "100_a1e1e678-4adc-420a-
bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-bdbe-54b81b3c25a6_5",
                    "type": "100",
                    "variant": "MANUAL-DSK",
                    "slug": "Mosart=L|M|O",
                    "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
                    "status": 1,
                    "graphicType": "MANUAL",
                    "handlerName": "DSK",
                    "description": "(DSK) - Mosart=L|M|O",
                    "hasContent": true,
                    "in": 0,
                    "duration": 5,
                    "plannedDuration": 5,
                    "actualDuration": 0,
                    "fields": [
                        {
                            "name": "graphics_description",
                            "value": "Mosart=L|M|O",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "graphics_id",
                            "value": "13059",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "tc_dur",
                            "value": "00:05",
                            "default": "00:00",
                            "fieldType": "TIMECODE",
                            "keyList": null,
                            "inputMask": "mm:ss",
                            "servers": null
```

```
                        },
                        {
                            "name": "continuecount",
                            "value": "-2",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "payloaduri",
                            "value": "/dataelements/13059/payload",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "thumbnailuri",
                            "value": "/dataelements/13059/thumb",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "elementuri",
                            "value": "/dataelements/13059",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "editoruri",
                            "value": "/app/pilotedge/pilotedge.html?
dataelement=13059&edit=",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        }
                    ],
                    "hasTemplate": null,
                    "emptyTemplate": null,
                    "templatePlaceHolders": null
                }
            ],
```

```
            "items": [
                {
                    "subItems": [
                        {
                            "id": "100_a1e1e678-4adc-420a-
bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-bdbe-54b81b3c25a6_2",
                            "type": "100",
                            "variant": "TIMECODE-DSK",
                            "slug": "Mosart=L|-00:00|00:29",
                            "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
                            "status": 0,
                            "graphicType": "",
                            "handlerName": "DSK",
                            "description": "(DSK) - Mosart=L|-00:00|00:29",
                            "hasContent": true,
                            "in": 0,
                            "duration": 29,
                            "plannedDuration": 29,
                            "actualDuration": 0,
                            "fields": [
                                {
                                    "name": "graphics_description",
                                    "value": "Mosart=L|-00:00|00:29",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "graphics_id",
                                    "value": "23757",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "tc_dur",
                                    "value": "00:29",
                                    "default": "00:00",
                                    "fieldType": "TIMECODE",
                                    "keyList": null,
                                    "inputMask": "mm:ss",
                                    "servers": null
                                },
                                {
                                    "name": "continuecount",
                                    "value": "0",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
```

```
                                "inputMask": null,
                                "servers": null
                            },
                            {
                                "name": "payloaduri",
                                "value": "/dataelements/23757/payload",
                                "default": null,
                                "fieldType": "TEXT",
                                "keyList": null,
                                "inputMask": null,
                                "servers": null
                            },
                            {
                                "name": "thumbnailuri",
                                "value": "/dataelements/23757/thumb",
                                "default": null,
                                "fieldType": "TEXT",
                                "keyList": null,
                                "inputMask": null,
                                "servers": null
                            },
                            {
                                "name": "elementuri",
                                "value": "/dataelements/23757",
                                "default": null,
                                "fieldType": "TEXT",
                                "keyList": null,
                                "inputMask": null,
                                "servers": null
                            },
                            {
                                "name": "editoruri",
                                "value": "/app/pilotedge/pilotedge.html?
dataelement=23757&edit=",
                                "default": null,
                                "fieldType": "TEXT",
                                "keyList": null,
                                "inputMask": null,
                                "servers": null
                            },
                            {
                                "name": "tc_in",
                                "value": "00:00",
                                "default": "00:00",
                                "fieldType": "TIMECODE",
                                "keyList": null,
                                "inputMask": "mm:ss",
                                "servers": null
                            }
                        ],
                        "hasTemplate": null,
                        "emptyTemplate": null,
                        "templatePlaceHolders": null
```

```
                        }
                    ],
                    "transition": {
                        "type": "Mix",
                        "rateOrIndex": 0
                    },
                    "bodyText": "",
                    "id": "0_a1e1e678-4adc-420a-bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-
bdbe-54b81b3c25a6_CAMERA1_1",
                    "type": "0",
                    "variant": "1",
                    "slug": "CAMERA 1",
                    "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
                    "status": 0,
                    "graphicType": null,
                    "handlerName": null,
                    "description": null,
                    "hasContent": true,
                    "in": 0,
                    "duration": 5,
                    "plannedDuration": 5,
                    "actualDuration": 0,
                    "fields": [
                        {
                            "name": "xpoint",
                            "value": "CAM1",
                            "default": "CAM1",
                            "fieldType": "LIST",
                            "keyList": "mixerinputs",
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "sr_create_story_clip",
                            "value": "False",
                            "default": null,
                            "fieldType": null,
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "sr_publish_story_clip",
                            "value": "False",
                            "default": null,
                            "fieldType": null,
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "sr_story_clip_id",
                            "value": "",
```

```
                            "default": null,
                            "fieldType": null,
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "sr_story_clip_slug",
                            "value": "",
                            "default": null,
                            "fieldType": null,
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        }
                    ],
                    "hasTemplate": true,
                    "emptyTemplate": false,
                    "templatePlaceHolders": null
                },
                {
                    "subItems": [],
                    "transition": {
                        "type": "Mix",
                        "rateOrIndex": 0
                    },
                    "bodyText": "",
                    "id": "4_a1e1e678-4adc-420a-bdbb-8aaa76dc6cf0_ba6f770a-1962-47ec-
bdbe-54b81b3c25a6_Mosart=F_4",
                    "type": "4",
                    "variant": "FULL",
                    "slug": "Mosart=F",
                    "storyId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
                    "status": 0,
                    "graphicType": null,
                    "handlerName": "FULL",
                    "description": null,
                    "hasContent": true,
                    "in": 5,
                    "duration": 5,
                    "plannedDuration": 5,
                    "actualDuration": 0,
                    "fields": [
                        {
                            "name": "scenename",
                            "value": null,
                            "default": "",
                            "fieldType": "TEXT",
                            "keyList": "",
                            "inputMask": null,
                            "servers": null
                        },
                        {
```

```
            "name": "text",
            "value": null,
            "default": "",
            "fieldType": "TEXT",
            "keyList": "",
            "inputMask": null,
            "servers": null
        },
        {
            "name": "graphics_description",
            "value": "Mosart=F",
            "default": "",
            "fieldType": "TEXT",
            "keyList": null,
            "inputMask": null,
            "servers": null
        },
        {
            "name": "graphics_id",
            "value": "23661",
            "default": null,
            "fieldType": "TEXT",
            "keyList": null,
            "inputMask": null,
            "servers": null
        },
        {
            "name": "tc_dur",
            "value": "00:05",
            "default": "00:00",
            "fieldType": "TIMECODE",
            "keyList": null,
            "inputMask": "mm:ss",
            "servers": null
        },
        {
            "name": "continuecount",
            "value": "1",
            "default": null,
            "fieldType": "TEXT",
            "keyList": null,
            "inputMask": null,
            "servers": null
        },
        {
            "name": "payloaduri",
            "value": "/dataelements/23661/payload",
            "default": null,
            "fieldType": "TEXT",
            "keyList": null,
            "inputMask": null,
            "servers": null
        },
```

```
                        {
                            "name": "thumbnailuri",
                            "value": "/dataelements/23661/thumb",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "elementuri",
                            "value": "/dataelements/23661",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        },
                        {
                            "name": "editoruri",
                            "value": "/app/pilotedge/pilotedge.html?
dataelement=23661&edit=",
                            "default": null,
                            "fieldType": "TEXT",
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        }
                    ],
                    "hasTemplate": true,
                    "emptyTemplate": false,
                    "templatePlaceHolders": null
                }
            ]
        },
        {
            "id": "ca6a3127-48a6-42e9-814b-a79ccafa8a12",
            "insertId": "ba6f770a-1962-47ec-bdbe-54b81b3c25a6",
            "storyDuration": 5,
            "storyPlannedDuration": 0,
            "storyBackTime": -1,
            "pageNumber": "",
            "slug": "Untitled",
            "accessories": [],
            "items": [
                {
                    "subItems": [
                        {
                            "id": "100_a1e1e678-4adc-420a-
bdbb-8aaa76dc6cf0_ca6a3127-48a6-42e9-814b-a79ccafa8a12_2",
                            "type": "100",
                            "variant": "DSK",
                            "slug": "A test/Mosart=L",
```

```
"storyId": "ca6a3127-48a6-42e9-814b-a79ccafa8a12",
"status": 0,
"graphicType": "",
"handlerName": "DSK",
"description": "(DSK) - A test/Mosart=L",
"hasContent": true,
"in": 0,
"duration": 5,
"plannedDuration": 5,
"actualDuration": 0,
"fields": [
    {
        "name": "graphics_description",
        "value": "A test/Mosart=L",
        "default": null,
        "fieldType": "TEXT",
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "graphics_id",
        "value": "14617",
        "default": null,
        "fieldType": "TEXT",
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "tc_dur",
        "value": "00:05",
        "default": "00:00",
        "fieldType": "TIMECODE",
        "keyList": null,
        "inputMask": "mm:ss",
        "servers": null
    },
    {
        "name": "continuecount",
        "value": "-2",
        "default": null,
        "fieldType": "TEXT",
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "payloaduri",
        "value": "/dataelements/14617/payload",
        "default": null,
        "fieldType": "TEXT",
        "keyList": null,
```

```
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "thumbnailuri",
                                    "value": "/dataelements/14617/thumb",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "elementuri",
                                    "value": "/dataelements/14617",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "editoruri",
                                    "value": "/app/pilotedge/pilotedge.html?
dataelement=14617&edit=",
                                    "default": null,
                                    "fieldType": "TEXT",
                                    "keyList": null,
                                    "inputMask": null,
                                    "servers": null
                                },
                                {
                                    "name": "tc_in",
                                    "value": "00:00",
                                    "default": "00:00",
                                    "fieldType": "TIMECODE",
                                    "keyList": null,
                                    "inputMask": "mm:ss",
                                    "servers": null
                                }
                            ],
                            "hasTemplate": null,
                            "emptyTemplate": null,
                            "templatePlaceHolders": null
                        }
                    ],
                    "transition": {
                        "type": "Mix",
                        "rateOrIndex": 0
                    },
                    "bodyText": "",
                    "id": "0_a1e1e678-4adc-420a-
bdbb-8aaa76dc6cf0_ca6a3127-48a6-42e9-814b-a79ccafa8a12_CAMERA1_1",
```

```
"type": "0",
"variant": "1",
"slug": "CAMERA 1",
"storyId": "ca6a3127-48a6-42e9-814b-a79ccafa8a12",
"status": 0,
"graphicType": null,
"handlerName": null,
"description": null,
"hasContent": true,
"in": 0,
"duration": 5,
"plannedDuration": 5,
"actualDuration": 0,
"fields": [
    {
        "name": "xpoint",
        "value": "CAM1",
        "default": "CAM1",
        "fieldType": "LIST",
        "keyList": "mixerinputs",
        "inputMask": null,
        "servers": null
    },
    {
        "name": "sr_create_story_clip",
        "value": "False",
        "default": null,
        "fieldType": null,
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "sr_publish_story_clip",
        "value": "False",
        "default": null,
        "fieldType": null,
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "sr_story_clip_id",
        "value": "",
        "default": null,
        "fieldType": null,
        "keyList": null,
        "inputMask": null,
        "servers": null
    },
    {
        "name": "sr_story_clip_slug",
        "value": "",
```

```
                            "default": null,
                            "fieldType": null,
                            "keyList": null,
                            "inputMask": null,
                            "servers": null
                        }
                    ],
                    "hasTemplate": true,
                    "emptyTemplate": false,
                    "templatePlaceHolders": null
                }
            ]
        }
    ]
}
```


Open Rundown

GET `/api/v1/command/open-rundown?id={rundown-id}` *(since v5.10.0)*

Load a specific rundown in Mosart.

`rundowm-id` : Either the `name` or `ID` of the rundown. Both can be found from the response of `/rundown` .

Reload Rundown

GET `/api/v1/command/reload` *(since v5.4.0)*

Reload the current rundown.

Take Next

GET `/api/v1/command/start-continue` *(since v5.4.0)*

Take the next item in the rundown. Has some optional URL parameters:

`option` : Default, Continue, Mix, Wipe, Effect

`rate` : The transition rate for mix/wipe (integer)

`effect` : The effect number (integer)

`delay` : The delay in milliseconds for continue (integer)

Skip Next

GET `/api/v1/command/skip-next` *(since v5.10.0)*

Skip the next item in the rundown.

Unskip Next

GET `/api/v1/command/un-skip-next` *(since v5.10.0)*

Unskip the last skipped item in the rundown.

Skip Next Sub Item

GET `/api/v1/command/skip-next-sub-item` *(since v5.10.0)*

Skip the next sub item in the rundown.

Unskip Next Sub Item

GET `/api/v1/command/un-skip-next-sub-item` *(since v5.10.0)*

Unskip the last skipped sub item in the rundown.

Start Rundown from Top

GET `/api/v1/command/start-from-top` *(since v5.4.0)*

Restart the rundown from the first story.

Set as Next

GET `/api/v1/command/set-as-next?storyid={story-id}` *(since v5.10.0)*

Set story as next.

Enable/Disable Rehearsal Mode

GET `/api/v1/command/rehearsal-mode?state={state}` *(since v5.4.0)*

Enable/disable/toggle Rehearsal mode.

`state` : true = enable, false = disable, no parameter = toggle.

Take a Template

GET `/api/v1/command/template?type={template-type}&variant={variant}&bus={preview|program}&insert={true|false}` *(since v5.4.0)*

Takes the template by its type and variant (name) as well as optionally, which bus (Preview or Program) and, for Preview bus, the *insert* option (if the template should be inserted in front of or overwrite an existing manual item). If an optional parameter is not used, the defaults are *program* (for bus) and *false* for insert.

Examples:

GET `/api/v1/command/template?type=Camera&variant=CAM1`
This will cut the *Camera* template variant *CAM1* directly to Program.

GET `/api/v1/command/template?type=Camera&variant=CAM1&bus=PREVIEW`

This will put the *Camera* template variant *CAM1* into Preview, replacing any manual item already in Preview.

GET `/api/v1/command/template?type=Camera&variant=CAM1&bus=PREVIEW&insert=true`

This will put the *Camera* template variant *CAM1* into Preview, inserted in front of the item currently in Preview, *not* overwriting any manual item in Preview.

Trigger a DirectTake Template

GET `/api/v1/command/directtake?number={number}` *(since v5.4.0)*

Trigger a given DirectTake template. You must provide the template *number*.

Trigger Control Command

GET `/api/v1/command/controlcommand/{command}?params…` *(since v5.10.0)*

Trigger a control command in Mosart.

See also:

- *Keyboard-based Control Command Categories* in the Viz Mosart User Guide (section Viz Mosart User Interface > Keyboard Shortcuts > Control Command Keys)

and

- *Command Values and Parameters* in section *Control Commands in Templates*.

Here is a list of the control commands supported by the API along with the parameters the command takes.

Control command	Parameters (case sensitive)
start_from_top	
switch_rehearsal_mode	**Action**=TOGGLE | ACTIVATE | DEACTIVATE If no Action parameter is passed, it will DEACTIVATE.
autotake	**Action**=TRUE | ACTIVATE | FALSE | DEACTIVATE | TOGGLE If no Action parameter is passed, it will TOGGLE
play_story	**Story Name**=string
videowallmode	**MixEffect**=string

Control command	Parameters (case sensitive)
release_background	**CueOnly**=TRUE | FALSE
overlay_graphics	**Render**=string **Action**=CONTINUE | TAKE_MANUAL_OUT | TAKE_ALL_OUT | TAKE_LAST_OUT | PRETAKE_NEXT | CLEAR | MACRO | TAKE_NAMED_OVERLAY **Parameter**=string **Value**=string
directtake	**Template**=number
accessories	
ncs	**Action=**START_STATUS | STOP_STATUS **Parameter**=string
rundown_ncs_resync	
user_message	**Message**=string
overlay_to_manual	**Target**=SELECTED | ONAIR | PREVIEW **Handlers**=string **TakeOutMethod**=AUTOMATIC | MANUAL **storyid**=string
gui	
asrunlog_event	**Action**=RemotePanelFunctions **Parameter**=string
switch_graphics_mirroring	**Action**=TOGGLE | ACTIVATE | DE-ACTIVATE
enable_graphics_mirroring	**Target**=BOTH | OVERLAY | FULLSCREEN **Action**=TRUE | FALSE
switch_genlock_mode	**Action**=TOGGLE | ACTIVATE | DEACTIVATE
engine_switcher	**Action**=INIT | PRESET_STYLE | GOTO_PREV_PRESET **PresetName**=string **Transition**=CUT | TAKE
autotrans	**MixEffect**=PP | PROGRAM | ME1 | ME2 | ME3 **Transitionrate**=string

Copyright © 2026 Vizrt

Control command	Parameters (case sensitive)
video_server_goto	**ServerChannel**=ONAIR | PREVIEW **frame**=number
dve	**Direction**=FORWARD | FWD | REVERSE | REV | BACKWARD
fullscreen_graphics	**Engine**=string **Action**=CONTINUE | MACRO **Parameter**=string
audio	**Action**=FADE_MANUAL | FADE_OUT_KEEPS | FREEZE_AUDIO | SET_LEVEL_2_PREVIEW | SET_LEVEL_2_ONAIR | FADE_DOWN_ALL_MAINS | FADE_DOWN_SECONDARY_AUDIO | FADE_UP_LAST_MAINS | FADE_UP_SECONDARY_AUDIO **Faderate**=number
light	**Scene**=number
sequence	**Action**=START | STOP | LOOP | STOPLOOP
take_server_to_program	**MixEffect**=PP | PROGRAM | ME1 | ME2 | ME3 **Transitionrate**=number
video_port	**Action**=PLAY_PAUSE | STOP | CUE | RECUE | SET_LOOP | PLAY_TAIL | CUE_TAIL **VideoPort**=string **Parameter**=string
get_player_status	**VideoPort**: string
weather	**Action**=PLAY | CONTINUE | GOTO_FIRST
graphicsprofile	**Name**=string
studiosetup	**Name**=string
set_aux_crosspoint	**Bus**=string **CrossPoint**=string
set_crosspoint	**MixEffect**=PP | PROGRAM | ME1 | ME2 | ME3 | ME4 **Bus**=A | B | C | D | KEY1 | KEY2 | KEY3 | KEY4 | KEY5 | KEY6 **CrossPoint**=string

Copyright © 2026 Vizrt

Control command	Parameters (case sensitive)
transition_type	**TransitionType**=MIX | WIPE | EFFECT | Default | Toggle | Cut | None **Value**=number
record	**Command**=PREPARE | START | STOP | STOP_AND_DETACH | DELETE | GETSOM **ClipName**=string **PortName**=string **Group**=string **Recorder**=string **Duration**=number **IgnoreClipNamePattern**=TRUE | FALSE
device_property	*For audio mixer device properties:* **Parameter1**=AUDIO **Parameter2**=string The key of the property you want to set. **Parameter3**=string The value of the property you want to set. *For camera control device properties (format depending on robotic camera system):* **Anchor**=string **Category**=string **Speed**=string
set_videoserver_salvo	**Salvo**=string
switch_videoserver_mirroring	
reset_overunder	
item_update_fields	Rather use the Remote Control API command "Update source item in preview or program". For description of this command, see the entry for ITEM_UPDATE_FIELDS in Control Commands in Templates | Command Values and Parameters.

12.4 Story Recorder

Story Recorder enables recording and clip generation from a running Viz Mosart-driven show. It provides frame-accurate tracking of operator Take next commands, recording of story takes and retakes, and maintains an Edit Decision List (EDL) arranged in order of the taken items. Story Recorder also controls video recorders to capture PGM outputs, including clean and dirty feeds, enabling clip generation at item, story, or show level. A single story, or a collection of rundown stories can be individually published as a clip or the entire show can be combined into a single, show-level output. Story Recorder supports multiple output capabilities, from exporting clips to online platforms, to generating EDL files for post-production and third-party integrations.

- Introduction to Story Recorder
- Story Recorder Highlights
- Definitions

12.4.1 Introduction to Story Recorder

Story Recorder mode expands Viz Mosart's traditional role as a tool for managing live productions, to also support *prerecorded* productions and story-level clip generation. The producer benefits from both the failsafe, timely execution of gallery equipment operations *and* an ability to *pause and backtrack* when an improvement to the ongoing recording of a show is identified.

Story Recorder supports two main production workflows and an additional EDL output. Each can be enabled independently using the Mosart central **Settings** menu.

Workflow	Level	Resulting Output
Story clip production and publishing	Story	Stories marked for recording in the NRCS are individually recorded and published as clips, either during a live broadcast or in an offline session.
Show clip production, with pause and retake	Show	Full show recording with ability to pause, retake, and backtrack. An EDL is broadcast to a Vantage transcoder, which stitches all segments into a final clip.
Mosart EDL generation	Item, Story, Show	Generates JSON-format EDL files locally for post-processing, transcoding, or third-party integrations.

Story Recorder has been designed with a flexible output architecture. For the full table of validated NRCS, recorder, and transcoder combinations for each workflow, see Currently supported configurations section.

Info: All equipment involved, including the Mosart server machine and the recording system, must be connected to the same genlock network and master clock.
In the *Viz Mosart Administrator Guide* see
- section Frame Accurate System Operations.
- section Story Recorder Setup for all other setup and configuration requirements.

> **Info**: Operating Viz Mosart in Story Recorder mode requires an additional license.
> Please contact Vizrt support.

The following sections introduce Story Recorder with examples of common production scenarios:

- Story Recorder Setup
- Working with Story Recorder
- Story Recorder Troubleshooting

12.4.2 Story Recorder Highlights

- **Frame accurate**
  Studio equipment operations are synchronized down to frame-level for a core set of devices (video switcher, video server, audio mixer). This enables precise cuts and transitions for the supported template features.
- **Multiple output options**
  Clip generation using Mimir or TriCaster, and EDL output in Vantage or Viz Mosart format. Multiple outputs can be active simultaneously.
- **NRCS integration for story-level recording**
  Stories can be marked for recording in the NRCS or rundown editor (Saga, Vizrt's Showmaker), and these are visualized directly in the Viz Mosart Rundown view, with a REC indicator.
- **Records both clean and dirty feeds**
  As well as the current show with all graphics (dirty), Story Recorder retains a clean feed that can for example, be rebranded, or reused in another language.
- **Pause, retake, and backtracking** *(Vantage EDL workflow only)*
  When producing a final show clip with Vantage, the recording can be paused at any point and resumed from a selected Retake point. Story Recorder tracks back along the timeline, restoring the state of graphics, audio, and other attributes. An automatic snippet generator provides a short review clip of the cut point after a retake, and video/audio devices plus graphics are reset to a known state before recording resumes.
- **Inbuilt file housekeeping**
  Recorded video files are always large. Viz Mosart offers an active purging regime to optimize disk space.
- **Timing information**
  Clear display of over/under countdown timers and accumulated record time of the show.
- **Future-proof design**
  Story Recorder has been purposefully designed with a flexible interface to third party systems like editors, transcoders and asset management.
- **Keyboard shortcuts and control commands**
  Several keyboard shortcut options and control commands are available for Story Recorder, to ease day-to-day operation. Story Recorder mode, for example, can be toggled on/off via the UI switch, a keyboard shortcut, or a template control command.

12.4.3 Definitions

To help you around the following sections, here are some terms used specifically within context of Story Recorder.

- **AsRunLog**: Viz Mosart's internal log of show recording events, with timestamp.
  Events are internal technical transactions, for example RUNDOWN_STARTED, RUNDOWN_ENDED, STORY_STARTED, STORY_ENDED.
- **Clean feed**: An SDI video signal that does not have any added (burned in) graphics or text.
- **Consolidation**: Stitching. The process that begins in the background with ongoing generating of an EDL file that holds all the timecodes of the Mosart primary events that were sent to PGM.
  The EDL assets (all recordings, including clips, cuts and graphics) are sent to a transcoder where they are rendered into a single clip.
- **Dirty feed**: The stream on PGM, program circuit or on-screen graphics (even when there is no commentary or atmospheric sounds).
- **EDL**: Edit Decision List. Video editing involves listing timecodes of specific frames at the point where video edits are to be made. These edit points form an edit decision list, used in the post-production process of the show or video editing.
  In the simplest case, these points are written on paper, but usually are stored as metadata in a file that can be used by an NLE or transcoder.
- **Frame**: The smallest measurement in a timecode, precisely specifying a single and complete video picture, comprising two interlaced fields.
- **Frame accuracy**: Ability to edit media and control associated studio equipment with timecodes defined to frame level, that will always trigger at the start of the specified frame. This enables seamless cuts and transitions, whilst ensuring predictable, synchronized equipment behavior.
- **GUI**: Graphical User Interface. In this context, usually the Viz Mosart client.
- **LTC**: Linear (or Longitudinal) Time Code, a timing signal (for synchronization) embedded in audio recordings.
- **NLE**: Non linear editor. An editing station, for example Adobe Premiere Pro.
- **NRCS**: Newsroom control system. Planning utility for show design. Can also refer to Vizrt's *Showmaker*.
- **Pause**: The operation of stalling recording of the current story row. For the Vantage workflow, this operation also initiates backtracking to the start of the interrupted story item (primary template), and insertion of a pre-roll event in front of this primary template.
- **Pre-roll**: A ten second (by default, but may be different on your setup) period that a camera (anchor) is given to prepare in a video production, before the production is live. At *pause*, this period is also used behind the scenes for the technical setup (studio equipment, clip players, graphics players, recorders and transition continuity etc.) to be made ready for the new take.
- **Pre-roll item**: A pseudo-template which, at pause, is inserted in front of the interrupted primary item of the executing story.
- **PRG**: Program
- **PRV**: Preview
- **Recorded segments**: The usable parts that get stitched together to build the final clip from the recorded show.
- **Rendering/Stitching**: A transcoding task where clips are joined to form a single file, based on cut points and metadata contained in the session's EDL.
- **Story**: A Viz Mosart equivalent of an NRCS-created MOS story. It covers a single topic (for example, an interview) comprising several items (for example, multiple camera scenes).
- **Story item**: A single element of a *story* (for example, a single camera scene) associated with a discrete Mosart template.

- **Timecode**: Exact position of a frame in a recording, used to locate a recorded point. Can be relative (to the start of the clip) or absolute (time of day of the recording) or based on another reference, such as studio clock (wall) time. Used to synchronize when referencing all audiovisual media, for example, video, audio, captions, visual effects, and more.
- **Timeline**: A chronological display of on-air events (for example, story items) showing their start time and duration. In the Story Recorder window, these are represented by rows.
- **Transcoding**: Digital-to-digital file format conversion from one standard of encoding to another. For example, a movie file encoded for the *PAL* system can be converted to *NTSC,* or mxf to mp4 video. Also audio files and character encoding (for example, *UTF-8* to *ISO/IEC 8859*).
- **VITC**: Vertical Interval Time Code, (pronounced *vit-see)* is a timing signal embedded in a video recording.

12.4.4 Story Recorder Setup

- Overview
  - Story clip production and publishing
  - Show clip production with pause and retake
  - Mosart EDL generation
- Supported Configurations
- Setup Checklist
  - 1. Plan
  - 2. Install
  - 3. Make the system frame accurate
  - 4. Configure the video server
  - 5. Configure Story Recorder
  - 6. Verify
- Configure Story Recorder
  - What takes effect when
  - Enable Story Recorder
  - Clip Outputs
  - Recordings Housekeeping
  - Other Settings
  - Related General Setting
- Configuring the Video Server
  - Media resource types
  - To configure video server recorder ports
  - Preparing TriCaster
  - Clip naming pattern of temporary recordings
- Keyboard Shortcuts
  - Control commands
  - Pause Timeline shortcut
- Failover
- Troubleshooting

This section describes how to set up and configure Story Recorder in Viz Mosart. All Story Recorder settings are managed through Viz Mosart Central Settings, which gives one place for every output workflow and its parameters.

Work through the Setup Checklist below. It is the spine of the setup: each step links to the section or page that describes it, and the notes call out the mistakes that have caused hard to diagnose problems in the field.

End user (Viz Mosart operator) documentation for Story Recorder is provided in Working with Story Recorder.

Overview

Story Recorder relies on several Viz Mosart server components working together:

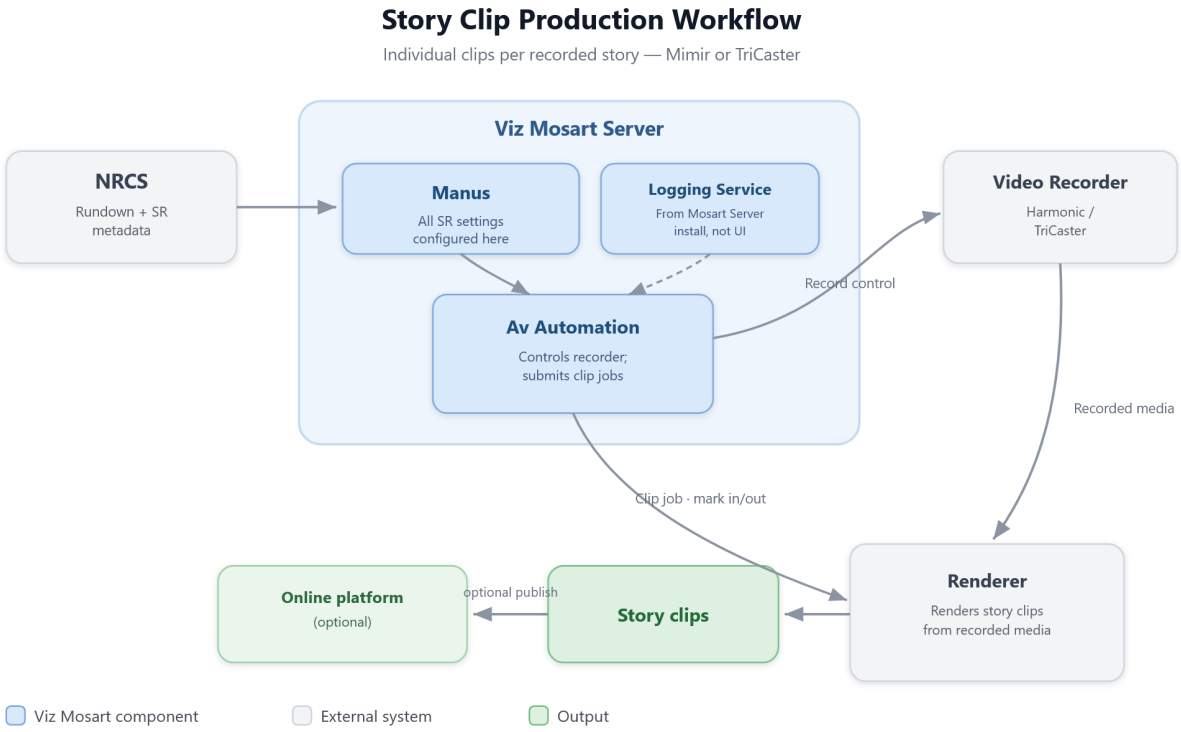
- **Manus Administrator** receives the rundown and story metadata needed for Story Recorder from the NRCS.
- **AV Automation** controls the video server (clip playout and record start and stop), handles communication with external renderers and transcoders, and uses genlock synchronized timecodes to ensure frame accurate clip boundaries and EDL entries.

- **Mosart Logging Service** (from the Mosart Server installation, not the UI) captures all on-air events with frame accurate timecodes. Story Recorder uses these events to build EDLs and determine clip in and out points.
- **Coder** (optional) generates short snippet clips around retake cut points for review in the Mosart UI.

Story Recorder produces three kinds of output, each targeting a different production need. The diagrams below show the data flow for each. See Supported Configurations for the devices validated with each one.

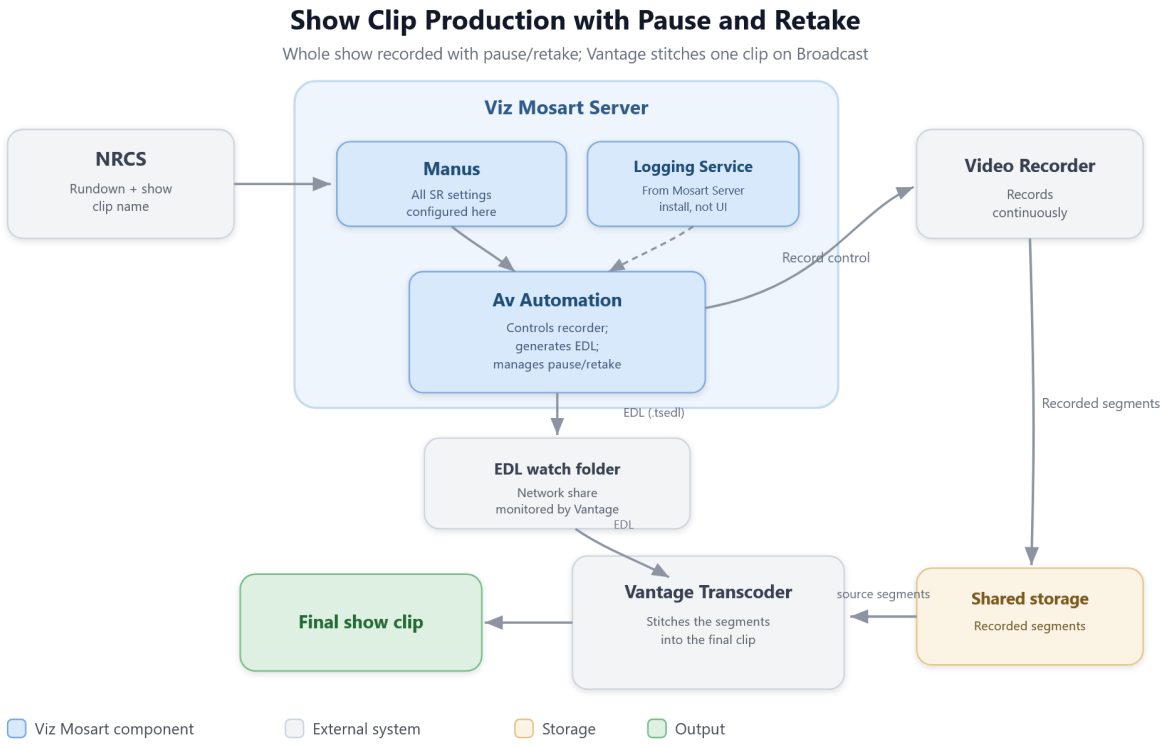
Story clip production and publishing

Stories marked for recording in the NRCS are individually recorded and published as clips, either during a live broadcast or in an offline session. See Story Clip Production Setup.



Show clip production with pause and retake

The entire show is recorded with the ability to pause, retake, and backtrack. At completion, an EDL is broadcast to a transcoder that stitches the recorded segments into a single final clip. See Show Clip Production Setup.



Mosart EDL generation

JSON format EDL files are generated locally, for post-processing, transcoding, or third-party integrations. Output level: *Item, Story, or Show*. See Mosart EDL Setup.

Supported Configurations

The table below lists the validated combinations of NRCS, recorder, and transcoder for each workflow. Story Recorder has been designed with a flexible output architecture, and new devices and integrations can be added as needs evolve.

Workflow	Output path	NRCS	Recorder	Transcoder/ Renderer	Online publisher	Product versions
Story clip production and publishing	TriCaster export	Any*	TriCaster	TriCaster	TriCaster	Viz Mosart 5.14+ TriCaster Vizion 8.4+
Story clip production and publishing	Mimir	Any*	Harmonic Spectrum	Mimir	-	Viz Mosart 5.14+ Mimir latest Saga latest

Workflow	Output path	NRCS	Recorder	Transcoder/ Renderer	Online publisher	Product versions
Show clip production with pause and retake	Network share (Vantage watch folder)	Any	TriCaster Harmonic Spectrum	Vantage Coder (optional, for snippet clip around cut point)	-	Viz Mosart 5.14+ TriCaster Vizion 8.4+ Viz Coder latest version
Mosart **EDL**	Local or network	Any	TriCaster Harmonic Spectrum	-	-	Viz Mosart 5.14+ TriCaster Vizion 8.4+

*An NRCS that can provide the required story metadata fields via the MOS protocol.

1. *Saga* offers built-in integration.
2. Other NRCS systems (for example, iNEWS) can be configured using Newsroom Settings field mapping. See Viz Mosart User Guide sections *Story Recorder Mode* for the MOS integration details.

Setup Checklist

Story Recorder setup spans several pages and applications. Work through the steps below before the first production. The **Applies to** column tells you which steps your workflow needs.

1. Plan

#	Step	Applies to	Where	Notes
1	Order the Story Recorder operating license from Vizrt Support.	All workflows	-	
2	Choose the output workflow: story clip, show clip, or Mosart EDL.	All workflows	Overview	The choice decides the recorder and the transcoder, so make it before you order or wire anything.
3	Check that your NRCS, recorder and transcoder are a validated combination, and that the product versions match.	All workflows	Supported Configurations	New devices and integrations can be added. Contact Vizrt if your combination is not listed.

#	Step	Applies to	Where	Notes
4	Plan the show design. Find out which templates and features work in frame accurate mode and with Story Recorder before you build the template set.	All workflows	Show Design for Frame Accuracy and Story Recorder	Do this before you build anything. The design decides which features you can use, and reworking a finished template set is expensive.

2. Install

#	Step	Applies to	Where	Notes
5	Install in the right order. When the Viz Mosart UI, Audio Player and Server share a machine, install the Server *last*, and point the Mosart Logging Service at the Server installation folder.	All workflows	System Logging	The Story Recorder adapter runs inside the Logging Service, on the server side.
6	Confirm that Viz Mosart runs the show correctly in live (non Story Recorder) mode.	All workflows	-	Story Recorder adds to a working system. Chasing a Story Recorder problem that turns out to be a plain playout problem wastes the most time of anything on this list.
7	Install Viz Coder on both the main and the backup server, if operators review the cut after a retake. Only the Coder *Worker* is needed.	Show clip	Show Clip Production Setup	Optional. Without it, retakes still work, but no snippet is produced for review.

3. Make the system frame accurate

Story Recorder can be switched on and tried out without genlock, but only for testing. In production it needs frame accuracy, and the steps below are what deliver it.

#	Step	Applies to	Where	Notes
8	Provide a clock source on the main and the backup server: a Plura timecode reader card, or PTP.	All workflows	Frame Accurate System Operations, PTP Setup	Both sources give the same frame accurate behavior. A Plura card needs a free slot in each machine; PTP needs a grandmaster on the network.
9	Switch on **Genlock mode** in *Viz Mosart Central Settings > AV Automation > Genlock*, and lock every device in the video chain. The Viz Mosart UI must show no genlock warning when Story Recorder is switched on.	All workflows	Frame Accurate System Operations	Without genlock, Viz Mosart falls back to its internal clock. That clock drifts, so clip boundaries degrade over the length of the show.
10	Confirm that the timecode reader delivers timecodes, not only frame lock. Switch on **Statistics on** in *Viz Mosart Central Settings > AV Automation > Genlock*, and check the analysis report and that `IsTcSyncedWithExternal = True` in the summary file.	All workflows	Frame Accurate System Operations	A system can be frame locked while the timecodes still come from the PC clock. Everything looks healthy, but every mark is offset by a constant amount. This is the most common field problem.
11	Set one time convention across every device that stamps timecodes. **Use local time** (Manus Settings > General) must match the master clock that feeds the video server, and **External clock is local time** (*Viz Mosart Central Settings > AV Automation > Genlock*) must describe the incoming signal. Set **Frame rate** (Manus Settings > General) to the rate the facility runs at.	All workflows	Frame Accurate System Operations, Related General Setting	A convention mix inside the timecode chain cannot be compensated. It is easy to recognize: every mark is wrong by the same amount, and that amount equals the difference between local time and UTC (5 hours when the local time zone is US Eastern, for example). Devices that do not stamp timecodes into recordings or EDLs can keep their own clocks. Local time also jumps with daylight saving, which is why many facilities run the house clock on UTC.

#	Step	Applies to	Where	Notes
12	Calibrate the controlled devices and enter the latencies in *Viz Mosart Central Settings > AV Automation > Genlock > Calibration*. Each latency accepts a value from 0 to 100.	All workflows	Device Calibration	Clicking **Apply** in AV Automation sends genlock and latency changes to Story Recorder immediately, so both always work with the same values.
13	Build the dedicated Story Recorder template set and test it in frame accurate mode.	All workflows	Show Design for Frame Accuracy and Story Recorder	Templates behave differently in frame accurate mode, where primary events are latency compensated, and not every feature from Standard mode is available. Do not reuse the live template set unchecked.

4. Configure the video server

#	Step	Applies to	Where	Notes
14	Configure the recording ports. Select **IsEnabled** in the **Story Recorder** section, set the file extension to match the server output (`mov` for TriCaster, `mxf` for Harmonic Spectrum), then reload the ports with **Ctrl+Shift+V**.	All workflows	Configuring the Video Server	Recordings are prepared on the record port of the current salvo. A running recording keeps the port it started on; a salvo change takes effect from the next recording.
15	Set the playout and recording directories. Show clip production needs the recording directory to be the *same* as the playout directory, because the preroll template plays back from the recording.	All workflows	Media resource types	Story clips are usually needed for playout, so they belong in a folder that Viz Mosart discovers clips in. Snippet clips do not.

#	Step	Applies to	Where	Notes
16	Prepare TriCaster: match its session frame rate to Viz Mosart, set its reference clock and its timecode source, and enable LTC.	TriCaster as recorder	Preparing TriCaster	Without LTC, TriCaster stamps the recording from the NDI recorder in UTC, and the export job has no reliable reference for the cut.
17	Verify the video server timecodes once during commissioning. Record about 30 seconds, then compare the first frame timecode of the recording file with the timecodes in the Mosart log. With Story Recorder enabled, no *Time sync issue* warning appears.	All workflows	Configuring the Video Server, Story Recorder Troubleshooting	Some video server models and firmware report a Start of Media that is offset from the actual recording. A constant offset here moves every produced clip by the same amount. The tolerance before the warning fires is 0.5 seconds on a synchronized external clock, and 1 second otherwise.

5. Configure Story Recorder

#	Step	Applies to	Where	Notes
18	Set **Enable Story Recorder** in Manus Settings.	All workflows	Settings (Manus) Configuration	The remaining Story Recorder settings sections become available only after this.
19	Configure the output for your workflow, with its mandatory folders and presets. Enabling Story Recorder must raise no configuration warnings.	All workflows	Story Clip Production Setup, Show Clip Production Setup, Mosart EDL Setup	With no output configured, recordings are made and nothing is published.

#	Step	Applies to	Where	Notes
20	Grant the Mosart Logging Service account read, write and delete permission on the recording folders, and read and write on every EDL and snippet folder. Verify with that account, not with an administrator login.	All workflows	Recordings Housekeeping and the workflow pages	An unreachable folder can block the whole enable action, and cleanup failures are reported only in the log.
21	Set the **Recording clip name prefix** and the **Recording retention days**.	All workflows	Recordings Housekeeping	Recordings are large files. Retention is 0 by default, which disables cleanup, so the disk fills unless you set a value or clean up by other means.
22	Configure the NRCS story metadata, so that stories marked for recording appear in the **Story Recorder** panel and the clip IDs sent by the NRCS resolve in the target system.	Story clip	Story Clip Production Setup	A story without valid metadata is tracked but never published. A stale clip ID fails the publish on every retry.
23	Set **Disable Story Recorder on rundown reload** deliberately, and brief the operators on what it does.	All workflows	Other Settings	With the default, a rundown reload ends the session and publishes the recording.
24	Assign the Story Recorder keyboard shortcuts.	All workflows	Keyboard Shortcuts	Show clip production needs a **Pause Timeline** shortcut before retakes can be used at all.
25	Save the settings while Story Recorder is off, and confirm that only one Mosart Logging Service instance is running.	All workflows	What takes effect when	A second service instance started from the same folder refuses to start.

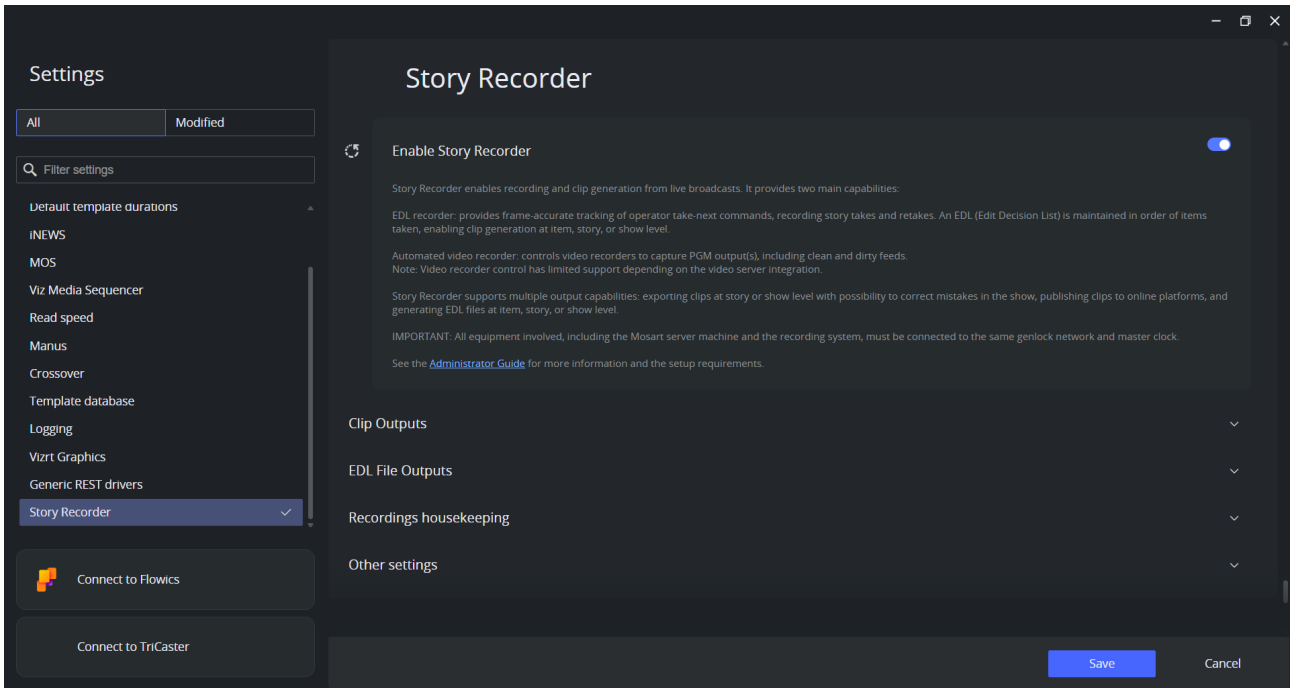
Copyright © 2026 Vizrt

6. Verify

#	Step	Applies to	Where	Notes
26	Configure Story Recorder identically on main and backup, set **Backup host**, **Backup port** and **Listener port**, and test one failover off air.	All workflows	Failover	Optional, and only for systems already set up with redundancy. An unreachable backup is reported only in the log; the session then runs unmirrored.
27	Run a complete off-air session: enable Story Recorder, record several stories, pause and retake if the workflow uses them, publish, and verify the output.	All workflows	Verification on Story Clip, Show Clip, Mosart EDL	Acceptance on the Mosart side is not proof. Confirm that the produced clip or EDL exists and has the expected content and duration.

Configure Story Recorder

Story Recorder output workflows and their parameters are configured through Viz Mosart Central Settings.



⚠ Save Story Recorder settings while Story Recorder is off. See What takes effect when for when a change is applied and what to restart.

What takes effect when

A small set of Story Recorder settings is read once, when the Mosart Logging Service starts. The rest are read when a session is enabled, so a save takes effect at the next enable and nothing needs restarting. Settings that live in AV Automation follow AV Automation's own rules.

You change	When it takes effect	What to restart
Enable Story Recorder, **Clip Outputs** (TriCaster, Mimir), the **Mosart** EDL output, and the dispatcher, backup and listener host and port settings	When the service starts	**Mosart Logging Service**, on the Mosart Server machine
Every other Story Recorder setting, including the whole **Vantage** EDL output, retakes, snippets, preroll, the clip name prefix and the retention days	At the next time Story Recorder is enabled	Nothing. Save while Story Recorder is off; a save during an active session is dropped and the service logs a warning.
Recording clip name prefix, **Recording retention days**	The housekeeping job runs at each service start, then at intervals	Nothing, unless you want the cleanup to run now, which needs the **Mosart Logging Service**
Use local time (Manus Settings > General)	Story Recorder picks it up at the next enable. AV Automation captures it once at startup.	**AV Automation**
Frame rate (Manus Settings > General)	Manus Administrator and the Viz Mosart UI apply it live. AV Automation caches it at startup.	**AV Automation**
Video server recorder ports, file extension, directories	On reconnect to the video server	**Ctrl+Shift+V** in AV Automation, or restart AV Automation
Genlock mode, **Clock Source**, calibration latencies (*Viz Mosart Central Settings > AV Automation > Genlock*)	On **Save**, and the values reach Story Recorder immediately	Nothing

You change	When it takes effect	What to restart
Plura card properties (*Viz Mosart Central Settings > AV Automation > Genlock*)	On **Save**	Nothing
`GenLogRepositoryConfig.xml`	Read once at service start	**Mosart Logging Service**
Newsroom Settings **Story External Metadata** mapping	After a restart	**Viz Mosart Server**

In Windows Services the service is registered as `MosartLogService` and displayed as **Mosart Logging Service**.

Enable Story Recorder

Setting	Default	Description
Enable Story Recorder	`false`	Enables the Story Recorder feature. When enabled, the remaining Story Recorder settings sections become available.

Clip Outputs

Configure clip output device types. Story clips are produced by the listed devices as configured here. The settings are specific to the output, and are described on the workflow pages:

- Story Clip Production Setup
- Show Clip Production Setup
- Mosart EDL Setup

Recordings Housekeeping

Automatic cleanup of temporary Story Recorder recording files on the video server.

Setting	Default	Description
Recording clip name prefix	`SRVizrt`	Prefix added to each intermediate recording clip that Story Recorder creates on the video server. Serves two purposes: - Identifies SR-generated recordings. - Acts as a filter for the housekeeping cleanup job. The clean-up job accesses the recording folders (configured per recorder port in AV Automation) as network shares. The *Mosart Logging Service* account must have **read, write, and delete permissions** on these folders.

Setting	Default	Description
Recording retention days	`0` days	Retention period for the automatic housekeeping task. Recordings matching the clip name prefix are deleted after the specified number of days. Set to *0* to disable housekeeping (default). The recording folders are automatically collected from the recorder port configurations in AV Automation. If defined, the housekeeping task is run every time *Mosart Logging Service* is (re)started and keeps running at regular intervals following this setting. Recorded clips can be *very large* files, easily filling the disk space.

Other Settings

Setting	Default	Description
Remote dispatcher host	`localhost`	Host address for connecting to the **Mosart Remote Dispatcher Service** (MosartRemoteControlService). This service dispatches commands between the Story Recorder Adapter and the Mosart server. The Mosart Remote Dispatcher service is always local, running on the Mosart server machine. *No other value should be given.*
Remote dispatcher port	`8090`	Port for connecting to the Mosart Remote Dispatcher Service.
Backup host	*(empty)*	Host address of the **backup Story Recorder Adapter** for failover. Leave empty for single-adapter mode (no failover). When configured, the primary adapter forwards its state to the backup, which can take over if the primary fails.
Backup port	`5001`	Port of any backup Story Recorder Adapter.
Listener port	`5001`	Port the Story Recorder Adapter listens on for incoming state from a primary adapter. Used in a primary/backup failover pair.

Copyright © 2026 Vizrt

Setting	Default	Description
Disable Story Recorder on rundown reload	`true`	Controls what happens when the NRCS rundown is reloaded during an active recording. • *Enabled* (default): SR mode is fully disabled, recordings stop and pending items are published. • *Disabled*: The timeline is only paused; SR mode stays active and can resume when the timeline starts again with the new rundown.

Related General Setting

The **Use local time** setting in the **General** section of Manus Settings applies to all Story Recorder timecodes:

Setting	Default	Description
Use local time	`false`	Chooses the time convention Viz Mosart uses for its own timecodes: local time when enabled, UTC when disabled. Set it to match the time reference of the master clock that feeds the video server. Story Recorder computes each mark as the event timecode minus the Start of Media reported by the video server, and that Start of Media is used exactly as delivered, so both values have to be in the same convention. A mismatch offsets every mark by the difference between local time and UTC. Calibration cannot absorb it, and with Story Recorder enabled Viz Mosart logs a warning that the video server time differs from the Mosart time. The setting covers every Story Recorder output: the Vantage and Mosart EDL files, and the render and export commands sent to Mimir and TriCaster. It is independent of the AV Automation setting **External Clock Is Local Time**, which describes whether the incoming timecode signal carries local time or UTC. When the two differ, Viz Mosart converts the incoming timecode to its own convention, so its own clock works with any combination. Mosart logs, UI, and Timing Display are not affected. Read more in section Frame Accurate System Operations.

Configuring the Video Server

The video server is used for playout of clips both during the show, and when recording. The setup involves configuring playout ports, recorder ports, and ensuring the correct directory and file extension settings for the various types of media resources involved.

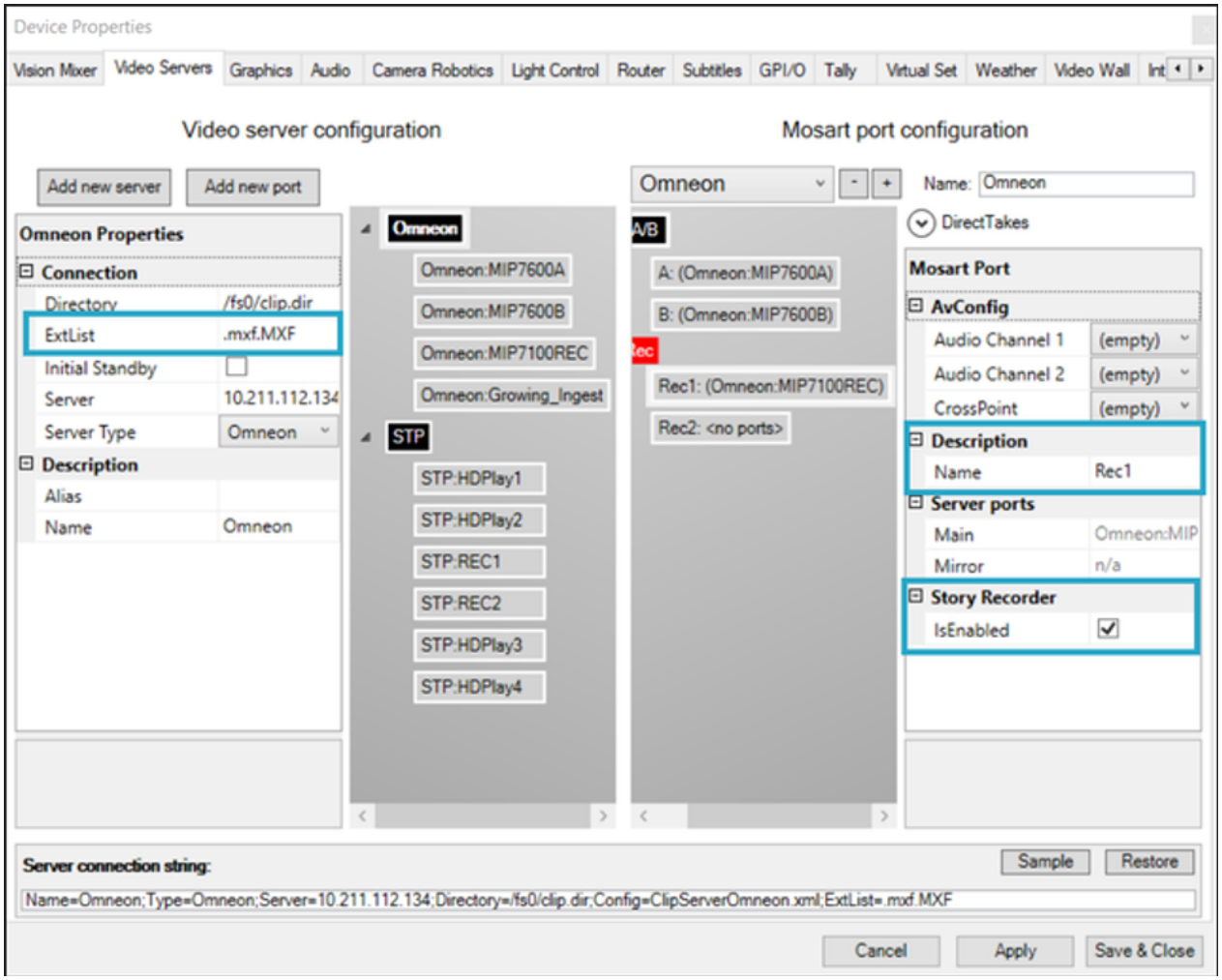
Media resource types

Story Recorder operations involve various types of media resources. Understanding these is important for correct directory and port configuration:

- **Playout clips**: The rundown clips monitored by Media Administrator, cued and played out by AV Automation. They must reside in a directory that the video server accepts for playout.
- **Recordings**: Clips produced when recording the video server output (typically the PGM output). In the ***story* clip production** workflow, these are the raw continuous recordings from which story clips are cut. These clips are not needed for playout unless the ***show* clip production** workflow is used, where the preroll template is played back from a recording clip. In this case, the **recording directory** must be *the same* as the **playout directory**.
- **Story clips**: Segments cut from the growing recordings, using an export job. Story clips are typically needed for playout, so they should reside in a folder that Viz Mosart discovers clips in.
- **Snippet clips**: Short review clips generated around retake cut points in the ***show* clip production** workflow. These are browsed in the **Story Recorder** floating window in the Viz Mosart UI, and used purely as visual verification of cut quality. They do not need to be available for playout.

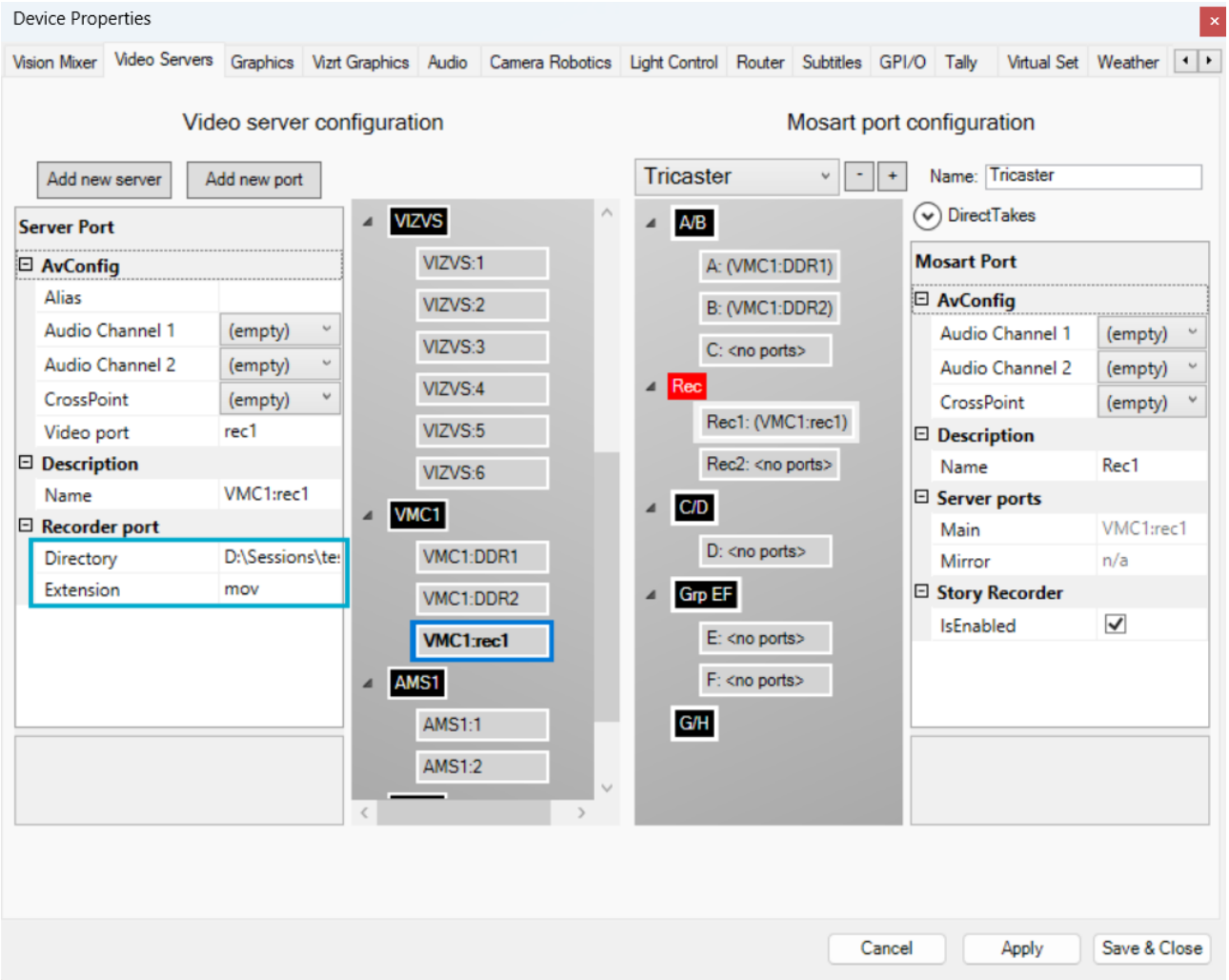
To configure video server recorder ports

1. In AV Automation, navigate to **Devices > Properties > Video Servers**.
2. The **Video server configuration** and **Mosart port configuration** should already be working for your system. Otherwise, refer to the Video Servers documentation.
3. Configure the recording port(s) dedicated for Story Recorder.
4. For the recording port(s), select **IsEnabled** in the **Story Recorder** section. This reserves the port for Story Recorder.



For TriCaster, the Quick Connect feature in Manus Settings can be used to automatically configure the video server setup and automatically enable the recorder for Story Recorder.

5. Set the recording file **extension** to match the format produced by the video server (for example, `mov` for TriCaster, `mxf` for Harmonic Spectrum).
   Story Recorder uses this extension to build the correct file path when referencing the recording clip in EDLs and render jobs. The extension is resolved with the following priority:
   a. **Extension** field on the recording port (per-port setting).
   b. `Ext` property in the **Server connection string**.
   c. `ExtList` property in the **Server connection string** (the first entry is used).
   d. If none of the above are set, the default is `mxf`.



6. By default, Mosart supports separate directories for playout clips and recordings (see screenshot above). However, for the ***show* clip production** workflow, the recording directory must be the same as the playout directory, because the preroll template is played back from the recording clip.
7. Click **Apply** then **Save & Close**.
8. Force a reconnect to the video server with **Ctrl+Shift+V** (or restart AV Automation).

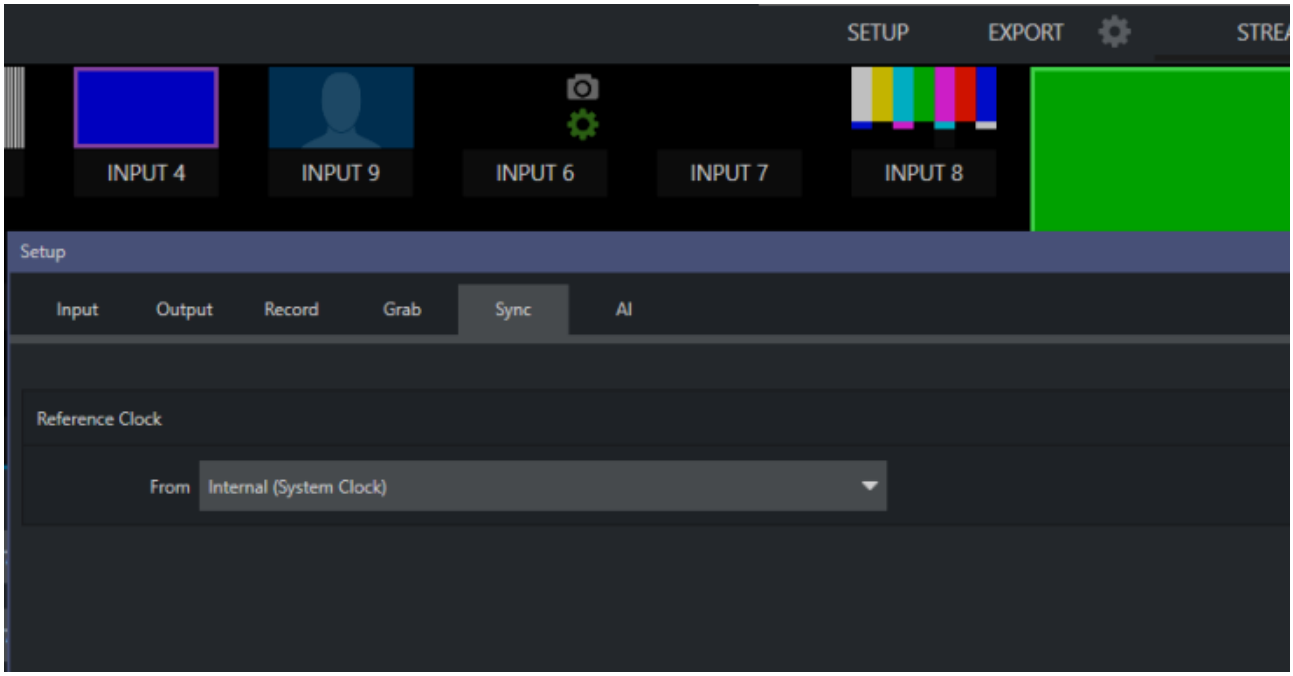
Preparing TriCaster

These steps apply whenever TriCaster acts as the recorder, in both the story clip and the show clip workflow. TriCaster stamps the recording with its own timecode, and Story Recorder cuts against that timecode, so TriCaster has to carry the same time as the rest of the system.

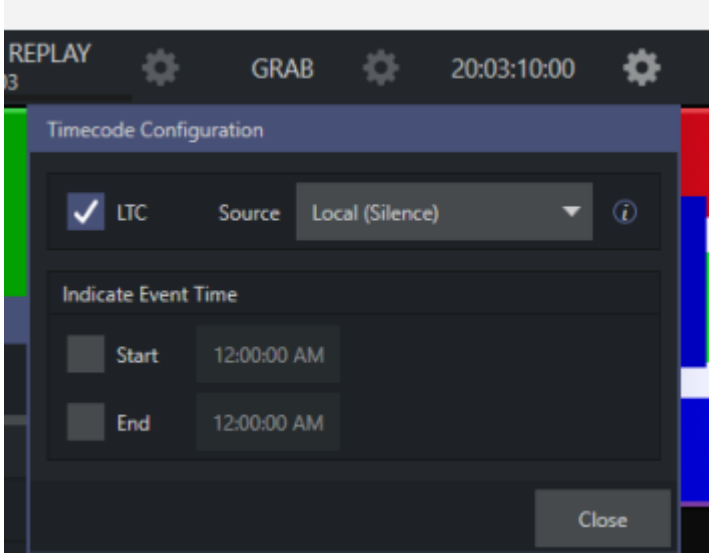
Two separate settings decide that, and they belong to different parts of TriCaster: the *reference clock* is what TriCaster locks its video to, and the *timecode source* is what it stamps the recording with. Set both. The menu paths below are from TriCaster Vizion; other TriCaster models arrange the same settings differently, so check the user guide for your model.

1. Set the TriCaster session to the same frame rate as Viz Mosart. The Story Recorder adapter calculates timecodes from the frame rate configured in Manus.

2. Set the reference clock. Open **Setup > Sync** and set **Reference Clock > From** to the house reference that the rest of the video chain is locked to. The default, **Internal (System Clock)**, means TriCaster free-runs on its own clock. Leave it on Internal only on a system that does not run frame accurate.



3. Set the timecode. Open **Timecode Configuration** and select **LTC**. For **Source**, select the house LTC input when the facility feeds one, or **Local (Silence)** when it does not. Local (Silence) makes TriCaster generate timecode from its own system clock rather than from an incoming signal.



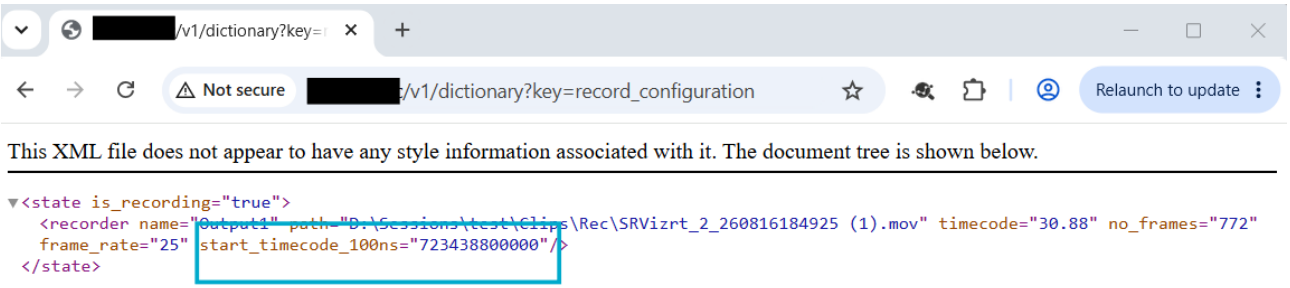
4. When you use **Local (Silence)**, synchronize the TriCaster and the Viz Mosart server clocks over NTP, and keep them within 1 second of each other. Story Recorder then works from time of day, and the two machines have to agree.
5. Configure TriCaster to record the program output (**Record Mix 1**) to a folder that Viz Mosart can discover clips in, as described below.

The two settings move together:

The facility has	Reference Clock (Setup > Sync)	Timecode source
House genlock and house LTC	The house reference	The LTC input
Neither	**Internal (System Clock)**	**Local (Silence)**, with both machines on NTP

Enable **LTC** in both cases, including on a system that does not run frame accurate. Without it, the start timecode of a recording comes from the NDI recorder and is UTC by default, which puts a whole-hour offset on every mark when Viz Mosart is configured for local time.

TriCaster can stop reporting Start of Media while it otherwise looks healthy: export jobs then fail or cut at the wrong point, and the Mosart log shows no Start of Media for the recording. Restart TriCaster, start the recording from TriCaster and verify this sends the start of media correctly http://tricaster-server/v1/dictionary?key=record_configuration.



For the reference and timecode connectors on your model, and the options in the **Reference Clock** list, see the TriCaster Vizion User Guide.

How Viz Mosart finds clips on TriCaster

Story clips are cut from the recording and are usually needed for playout, so the recording output folder has to suit the way Viz Mosart discovers clips. There are two routes, and only one of them is needed:

Route	Set up by	What the recording folder must be
TriCaster API	TriCaster Quick Connect, which writes the Media Administrator server connection and its HTTP credentials.	A folder TriCaster lists in its own file browser. Media Administrator queries TriCaster directly and polls the file browser for changes.
Media Service	Manual configuration of a Media Service publishing point in Media Administrator.	The folder that Media Service monitors. Using one folder for playout clips, recordings and story clips is the simplest arrangement.

TriCaster server connection string properties

Set these in the Server connection string in **AV Automation > Devices > Properties > Video Servers**.

Property	Default	Description
`HttpUsernam e`	`admin`	HTTP username for the TriCaster REST API. Used by all workflows. Mosart needs it to request Start of Media (the timecode of the first frame) for the render job command. Can be added via the *TriCaster Quick Connect* editor.
`HttpPasswor d`	*(empty)*	HTTP password for the TriCaster REST API, used together with `HttpUsername`. Can be added via the *TriCaster Quick Connect* editor.
`AddExportJob DelayMs`	`5000`	Delay in milliseconds before sending the export job request to TriCaster. Used by ***story* clip production** only. When a story ends, Mosart sends an export job to TriCaster to cut the story clip from the ongoing recording. If the export request is sent too soon (within 1 to 2 seconds of the cut point), TriCaster may not be ready to process it because the recording file is still growing. This delay gives TriCaster time to finalize the recording segment before the export is triggered.

Example: `...;HttpUsername=admin;HttpPassword=admin;AddExportJobDelayMs=5000`

Clip naming pattern of temporary recordings

The temporary recording files generated by Viz Mosart are automatically named.

Single recorder:

```
<prefix>_<counter>_<timestamp>.<ext>
```

Multiple recorders:

```
<prefix>_<recorder name>_<counter>_<timestamp>.<ext>
```

Where:

- **prefix**: The **Recording clip name prefix** configured in Manus Settings (default: `SRVizrt`). This prefix is also used by the housekeeping job to identify SR-generated recordings.
- **counter:** A sequence number, reset to 1 for each Story Recorder session.
- **timestamp:** An automatically generated (format `YYMMDDHHmmss`).
- **ext:** The file extension configured in **AV Automation > Devices > Properties > Video Servers > ExtList** (the first in the list is used).
- **recorder name:** The **Name** setting on the virtual recorder port in AV Automation (for example, `Rec1`). Only included when multiple recorders are configured.

Example: A single recorder, prefix `SRVizrt`

File
`SRVizrt_1_211001153956.mxf`
`SRVizrt_2_211001153956.mxf`
`SRVizrt_3_211001153956.mxf`

Example: Two recorders (`Rec1`, `Rec2`), prefix `SRVizrt`

File
`SRVizrt_Rec1_1_211014172330.mxf`
`SRVizrt_Rec1_2_211014172330.mxf`
`SRVizrt_Rec1_3_211014172330.mxf`
`SRVizrt_Rec2_1_211014172521.mxf`
`SRVizrt_Rec2_2_211014172521.mxf`
`SRVizrt_Rec2_3_211014172521.mxf`

Copyright © 2026 Vizrt

Keyboard Shortcuts

Control commands

Several Story Recorder actions are available for keyboard assignment. They are configured as **Control Command** shortcuts with the command `STORY_RECORDER` and an **Action** parameter:

Action	Applies to	Description
TOGGLE	All workflows	Toggles Story Recorder mode on or off.
ENABLE	All workflows	Enables Story Recorder mode.
DISABLE	All workflows	Disables Story Recorder mode and publishes the recorded clip(s).
DISABLE_NO_PUBLISH	All workflows	Disables Story Recorder mode and discards the session. Recording stops and nothing is published. This is the counterpart of DISABLE, which publishes.
PAUSE	Show clip	Pauses the timeline while in Story Recorder mode.
RETAKE	Show clip	Sets a retake point at the selected position.
BROADCAST_TIMELINE	Show clip	Delivers the show recorded so far to the configured outputs, the same as the **Deliver** button in the Story Recorder tool window (only available while paused).
REDO	All workflows	Starts the show again from the story that is on air. Story Recorder is disabled without publishing, the rundown is reloaded, and the story that was on air is set as the first story after the reload. Mainly used in show clip production.

Refer to the *Viz Mosart User Guide*, section *Viz Mosart User Interface > Keyboard Shortcuts > Control Command Keys*.

Pause Timeline shortcut

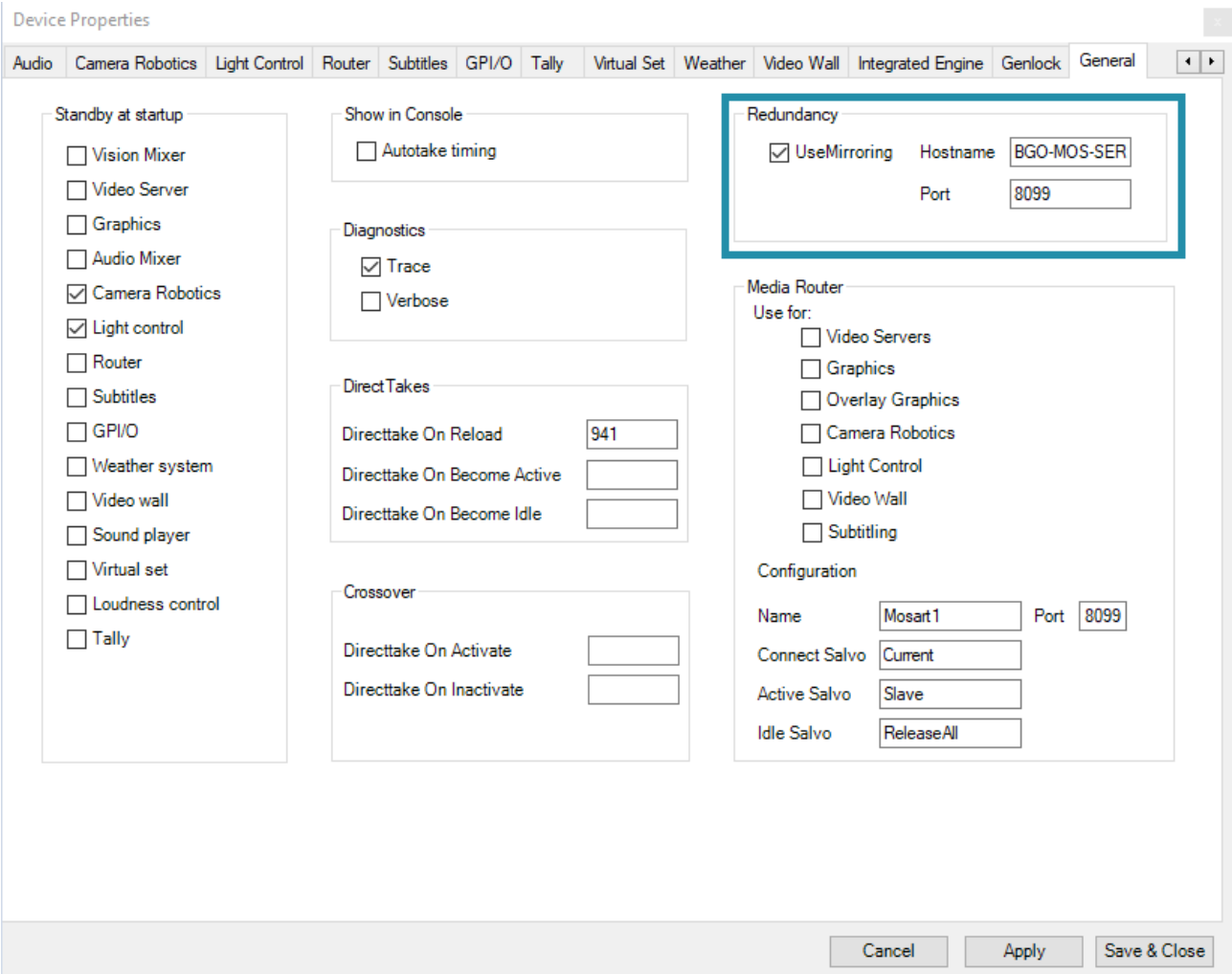
In addition to the control command actions, a dedicated **Pause Timeline** shortcut is available:

Shortcut name	Command	Applies to	Description
Pause Timeline	`pause_timeline`	Show clip	Pauses or resumes the Mosart timeline when in Story Recorder mode.

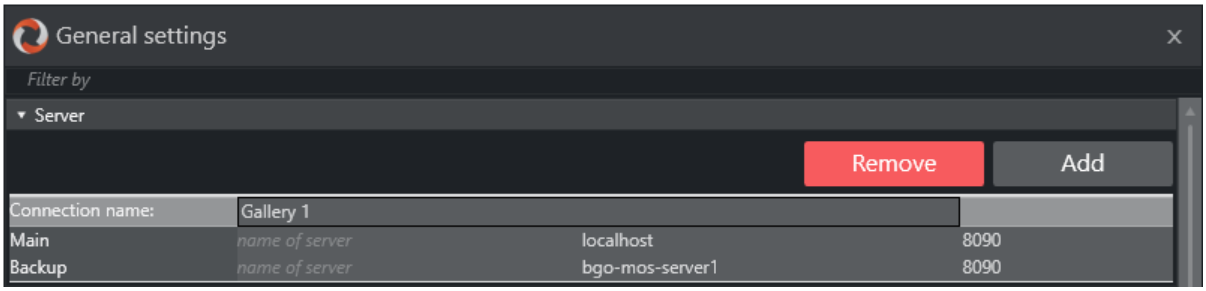
Failover

This step is optional and requires that Viz Mosart is already set up with redundancy as described in section Redundancy.
This means that:

- Both the Viz Mosart main and backup server have same configurations for at least Mosart Administrator, AV Automation, Media Administrator and Overlay Graphics Interface.
- AV Automation is configured for mirroring on both main and backup servers (with main pointing to backup and backup pointing to main):



- The Viz Mosart UI is configured to connect to the backup server:



For Story Recorder, there are additional steps:

- Story Recorder is configured on both main and backup in the same way in Manus Settings.
- **Backup host and port**: Configure the **Backup host** and **Backup port** settings in Manus Settings (under **Other Settings**) on the primary adapter, and the **Listener port** on the backup adapter.
- **Snippet clips folder (optional)**: If snippet generation around the cut point has been set up for main, set it up in the same way for backup. Install Viz Coder on *both* main and backup server. The snippet folder configured in the Settings menu must point to the same shared location on both main and backup, so that the Viz Mosart UI has access to snippet files regardless of which system is active.
- **Genlock calibration**: The same calibration settings from *Viz Mosart Central Settings > AV Automation > Genlock* must be configured on both main and backup to ensure frame-accurate timecodes are consistent across failover (see Frame Accurate System Operations).

Troubleshooting

If Story Recorder does not behave as described, see Story Recorder Troubleshooting.

Show Clip Production Setup

- Show Clip Production Setup
  - What you need
  - Settings
    - EDL File Outputs > Vantage
  - Procedure
    - Configure the video server ports
    - Configure the Vantage EDL output
    - Install Viz Coder for snippet generation
    - Set the preroll duration
    - Configure automatic naming of the final clip
    - Set automatic pause at end of show
    - Assign keyboard shortcuts
  - Clip naming pattern for EDL files and final show clips
  - Verification

Show Clip Production Setup

This workflow records a complete show while the operator pauses, retakes, and backtracks story items. At the end of the show, Viz Mosart writes an EDL (Edit Decision List) to the folder configured for the output, listing the recorded segments that make up the valid show, with the file, in-point and out-point of each. A transcoder picks the file up and renders the final show clip from those segments.

The EDL format is set per output. Viz Mosart currently supports two: **Vantage**, a Telestream `.tsedl` file written to a watch folder that Vantage monitors, and **Mosart**, a JSON EDL written to a local folder, used mainly for troubleshooting and described in Mosart EDL Setup. Further formats may be added in future releases. This page describes the workflow end to end with Vantage as the transcoder.

This page covers **step 19** of the Story Recorder Setup Checklist, together with **step 7** if you want snippet review. Work through steps 1 to 18 first: they cover the license, the installation order, the frame accuracy setup, the show design, and the video server.

> Story Recorder needs frame accuracy in production. You can configure and test this output without genlock, but the marks will not be frame accurate: the in and out points no longer match the recording frame for frame. See steps 8 to 13 of the Setup Checklist.

What you need

On top of the general Story Recorder setup, this output needs:

- At least **2 playout ports** (for A/B playout of show clips and preroll recordings) and **at least 1 recording port** (2 recording ports can be used for example for clean and dirty feeds).
- A shared network watch folder that both Viz Mosart and Vantage can reach.
- A local EDL backup folder on the Mosart Server machine.
- **Viz Coder** (optional) generates short snippet clips around retake cut points for review in the Mosart UI. Only the Coder *Worker* is needed.

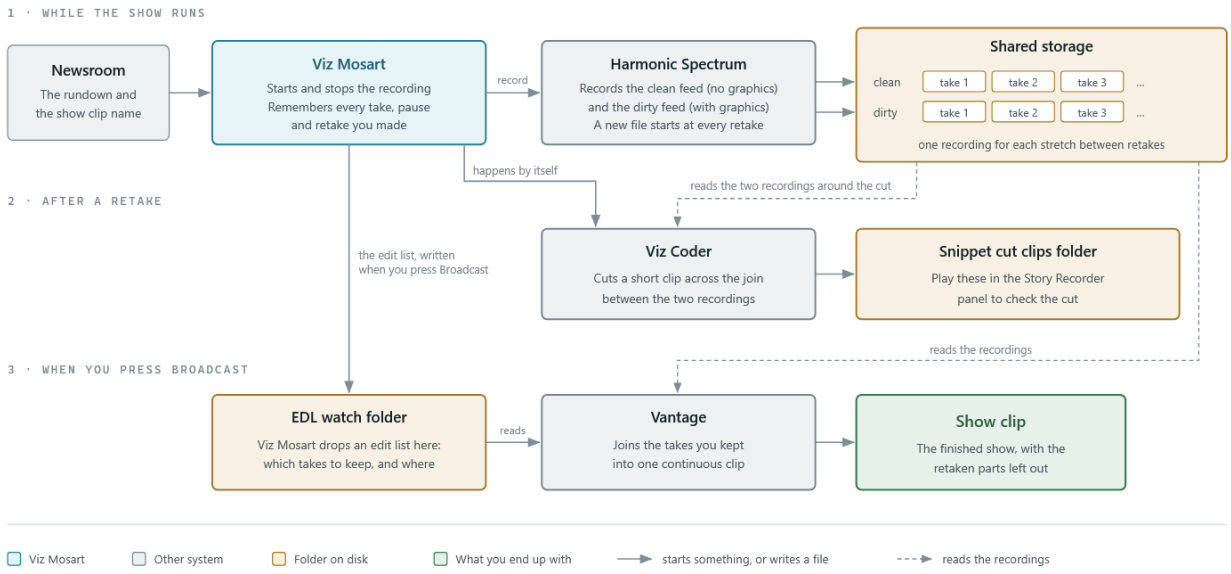
- A default media player for `*.mxf` files (for example, VLC) on the machine that runs the Viz Mosart GUI, so operators can open snippet clips.

> The folder where the final show clips reside is not controlled by Viz Mosart. This is set on the EDL transcoder side, here *Vantage*.

For the validated combination of NRCS, recorder and transcoder, and the product versions, see **Supported Configurations** in Story Recorder Setup.

Example: Vantage workflow with Harmonic Spectrum, clean/dirty recording, and snippet review

This diagram shows a specific deployment of the show clip production workflow using Harmonic Spectrum as video server, Vantage as transcoder, and Viz Coder for snippet generation around retake cut points. This configuration requires at least 2 playout ports (for A/B playout of show clips and preroll recordings) and 2 recording ports (for clean and dirty feeds).



Settings

Open the Manus Settings menu and set **Enable Story Recorder** to `true`. The remaining Story Recorder settings sections then become available.

> Save while Story Recorder is off, and restart the **Mosart Logging Service** after changing the **Clip Outputs** devices. See step 25 of the Setup Checklist.

EDL File Outputs > Vantage

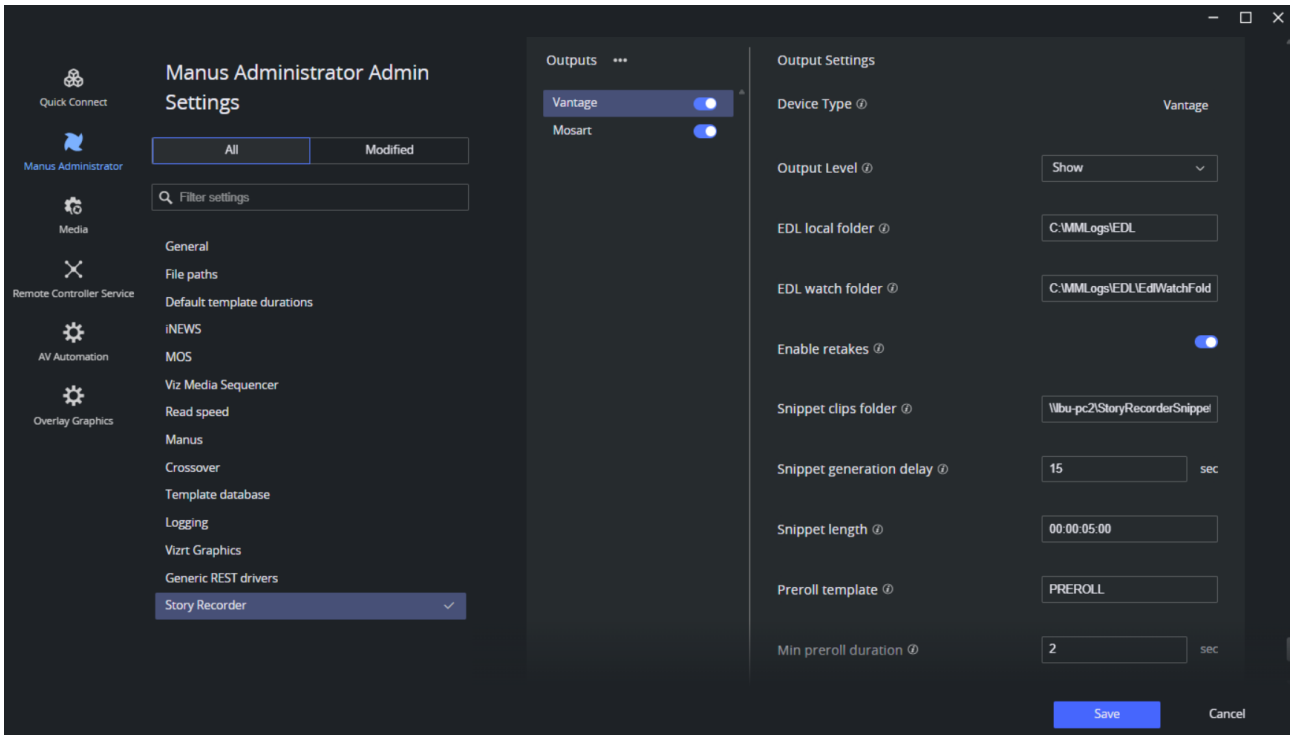
Produces `.tsedl` (Telestream EDL) files consumed by a Vantage transcoder for show-level clip stitching.

Setting	Default	Description
Output Level	Show	Read-only. Determines the granularity at which EDL files are produced. On *Show* level, one EDL is generated per show. The transcoder stitches recorded segments into a single show clip. Currently, only show-level EDL output is supported.
EDL local folder	`C:\MMLogs\EDL`	Local backup folder for `.tsedl` files. Files are written here simultaneously as to the Watch folder. The EDL transcoding node does not have access to this folder, but if ever the EDL files fail to be generated in the **EDL watch folder** because of say, connection issues, the user can later manually copy these files to the EDL transcode node. This setting cannot be empty and must be a separate location to **Edl watch folder** (for example, `C:\MMLogs\EDL` or `\\mosart_server_hostname\MMLogs\EDL`). This folder must have read and write permissions for the *Mosart Logging Service*. Viz Mosart does *not* clean up any files created in this folder.
EDL watch folder	*(empty)*	Shared network folder (UNC path, e.g. `\\vantage-server\watchfolder\`) monitored by the Vantage transcoder. When the show ends, the `.tsedl` file is written here. Vantage detects the file and automatically triggers clip assembly. This setting can not be empty and can be any shared network path where both Viz Mosart and the EDL transcoder have access (for example, `\\somehostname\EDL`). The folder must have read and write permissions for the *Mosart Logging Service*. Viz Mosart does *not* clean up any files created in this folder.
Enable retakes	`true`	Requires a **Pause Timeline** keyboard shortcut defined in the Viz Mosart UI that enables Pause and Retake during recording. When enabled, the operator can pause the timeline and re-record segments. The EDL instructs the transcoder to stitch only the valid segments. If disabled, the retake, preroll and pause controls are greyed, and a pause is refused with the warning *Pause ignored: retakes are switched off in the EDL output settings.*

The following settings are visible only when **Enable retakes** is on.

Setting	Default	Description
Snippet clips folder	*(empty)*	When a retake of a story item is performed, a rendered snippet around the cut point is placed in this folder shortly after the item is retaken. If this property is not configured or left empty, *no* snippet of the cut is created. The location must be a network shared folder where the user running *Mosart Logging Service* has read and write access and the user running *Mosart GUI* has at least read access. (for example, *\\somehostname\StoryRecorderSnippetClips*) To set up a snippet recorder see Install Viz Coder for snippet generation. Default: *empty*. This implies that a snippet recorder is optional.
Snippet generation delay	`15` sec	How long in seconds after the retake process started, that generation of the snippet clip (from around the cut) shall begin. The retake process is the last part from the following chain of actions: *Show is paused, pre-roll is taken on-air, pre-roll auto-takes the item to be retaken.* The default is the minimum recommended value.
Snippet length	`00:00:05:00`	Duration of *each* recording segment to be included in the snippet (format `HH:MM:SS:FF`). For example, with `00:00:05:00`, the snippet contains the last 5 seconds of the previous segment and the first 5 seconds of the new segment at the cut point.
Preroll template	`PREROLL`	The variant name of the Mosart template used for the preroll countdown between retakes. This template must exist in the *ChannelTemplates* file. When a retake starts, this template is played to allow the presenter time to recompose before the recording resumes. • If another Package template with this same name exists on your system which was manually created, *delete* or *rename* it. • The Preroll template for Story Recorder must retain its default minimal characteristics and can not contain additional control parameters. • This unique minimal template can be reused after an upgrade.
Min preroll duration	`10` sec	Minimum fixed duration of the preroll template. The preroll duration can be increased by the user in the Story Recorder panel. This time needs to be sufficient for the video server to reset and be ready to record, after being paused. Typically *10-15* seconds.

Setting	Default	Description
Max preroll duration	`30` sec	Maximum duration of the preroll template. The actual preroll duration is determined by the system within this range.



Procedure

Configure the video server ports

See **Configuring the Video Server** in Story Recorder Setup. This workflow needs the recording directory to be the same as the playout directory, because the preroll template plays back from the recording. With TriCaster as recorder, also work through **Preparing TriCaster** on the same page.

Configure the Vantage EDL output

1. In the Manus **Settings** menu, enable **Enable Story Recorder**.
2. Under **EDL File Outputs**, open the **Vantage** section.
3. Set **EDL local folder**. Keep `C:\MMLogs\EDL` or choose another path, and make sure it differs from the **EDL watch folder**. Grant the *Mosart Logging Service* account read and write permission on it.
4. Set **EDL watch folder** to the UNC path that Vantage monitors, for example `\\vantage-server\watchfolder\`. Grant the *Mosart Logging Service* account read and write permission on it.
5. Leave **Enable retakes** on if operators need to pause and re-record segments, and define a **Pause Timeline** keyboard shortcut in the Viz Mosart UI.

6. Configure the retake settings: **Snippet clips folder**, **Snippet generation delay**, **Snippet length**, **Preroll template**, **Min preroll duration**, and **Max preroll duration**.
7. Save the settings while Story Recorder is off. They take effect when the next session is enabled.

Install Viz Coder for snippet generation

This setup is optional. It generates a few seconds (configurable) of verification video, intended to reveal the integrity of a fresh cut, after a story item has been paused and then retaken. Snippet creation employs Vizrt's transcoder, Coder.

1. Download the most recent .exe version of Coder from the customer FTP, by navigating to `/products/VizOne/Latest Versions/Individual Installers and Docs/Coder/Latest Version`.
2. You need only install the Coder *Worker*. Install Coder on the same machine as the Viz Mosart Server. Coder's default installation location is `C:\Program Files\Vizrt\Coder`.
3. On the machine that runs the Viz Mosart GUI, define a default media player for `*.mxf` files (for example, VLC).
4. In the Manus **Settings** menu, for *Vantage*, configure **Snippet clips folder**, **Snippet generation delay**, and **Snippet length**. Save while Story Recorder is off.
5. Restart the **Viz Mosart UI**. The snippet settings take effect when the next session is enabled.

> If you install Coder anywhere but the default location, you must then update the configuration file `C:\Program Files (x86)\Mosart Medialab\Mosart Server\CoderTools.dll.Config` to specify the new path to the Coder executable, `amlbatch.exe`.

```xml
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <sectionGroup name="userSettings" type=
        "System.Configuration.UserSettingsGroup, System, Version=4.0.0.0,
        Culture=neutral, PublicKeyToken=b77a5c561934e089">
            <section name="CoderTools.Properties.Settings" type=
            "System.Configuration.ClientSettingsSection, System, Version=4.0.0.0,
            Culture=neutral, PublicKeyToken=b77a5c561934e089" allowExeDefinition=
            "MachineToLocalUser" requirePermission="false"/>
        </sectionGroup>
    </configSections>
    <userSettings>
        <CoderTools.Properties.Settings>
            <setting name="ExecutableFileLocation" serializeAs="String">
                <value>C:\Program Files\vizrt\Coder\bin\amlbatch.exe</value>
            </setting>
        </CoderTools.Properties.Settings>
    </userSettings>
<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8"/>
</startup></configuration>
```

Set the preroll duration

The pre-roll period must be long enough for the video server hardware to reset itself and be ready to continue recording, after a pause.

1. Determine the time required by your video server hardware to commence recording. This is typically 10-15 seconds.
2. Configure this value in **Min preroll duration**.
3. Operators may increase the value in the Story Recorder panel. The setting is retained between Story Recorder sessions.

Configure automatic naming of the final clip

This step is optional. By default, Viz Mosart names the show, its EDL files and the final clip after the rundown. It lets the NRCS supply the name instead, so the final clip arrives with the name the newsroom already uses for the show.

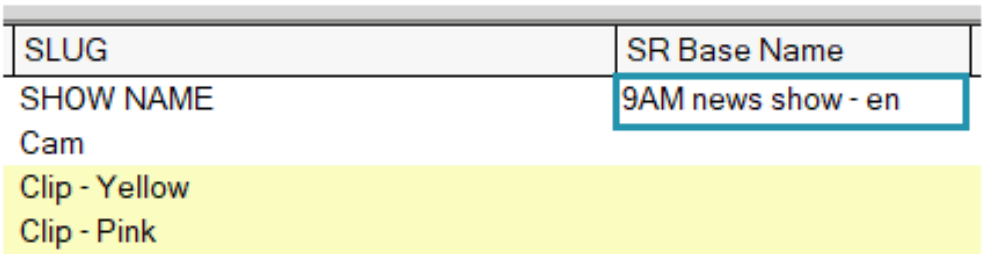
Three sources can provide the name. The first one that applies wins.

Source	When it is used
Name entered by the operator in the Story Recorder panel	Always. It overwrites the name from the NRCS.
NRCS field mapped to `sr_final_clip_name`	When the operator has not entered a name. This is what the steps below set up.
Rundown name	The default, when neither of the above is provided.

The setup has two sides: a column in the NRCS that carries the name, and a mapping in Viz Mosart that tells it which column to read.

In the NRCS:

1. Open the rundown for the show.
2. Add a column to hold the name, here *SR Base Name*. The procedure is specific to your NRCS and may require administrator privileges.
3. In the **first story** of the rundown, enter the name of the show, here *9AM news show - en*.



⚠ Viz Mosart reads the name from the first story of the rundown only. A name placed in any other story is ignored.

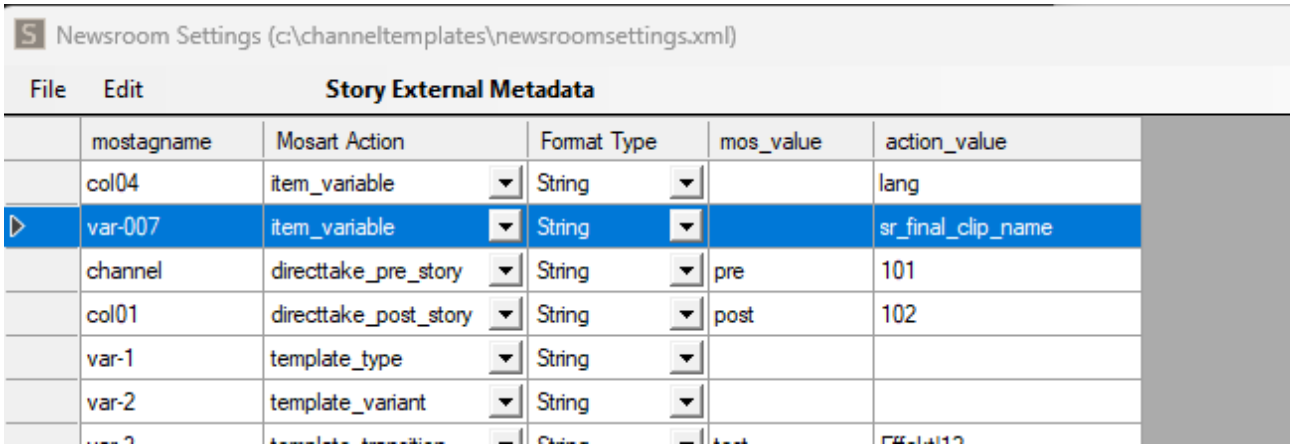
> **Tip.** If your NRCS works on rundown level rather than story level, OpenMedia for example, write a script that copies the rundown field into every story. That fills the first story as well, so Viz Mosart finds the name.

> **Tip.** If the first story of the rundown changes often, start the rundown with a *Break* that is not part of the show, and keep the name there. The name then survives changes to the running order.

In Viz Mosart:

1. In the Viz Mosart console application *Manus Administrator*, type `ns` to open the Newsroom Settings menu.
2. Select **Edit > Story External Metadata** and add a row with the values below, if it is not there already.

Field	Value
mostagname	The name of the column you added in the NRCS. In this iNEWS example, `var-007`.
Mosart Action	`item_variable`
Format Type	`String`
mos_value	Leave empty.
action_value	`sr_final_clip_name`



> The column name to enter as **mostagname** is not the label you see in the NRCS. To find the value your NRCS sends, here `var-007`, see the *Story External Metadata* section of Newsroom Settings Editor.

3. Restart the Viz Mosart Server.

Check that the name arrives:

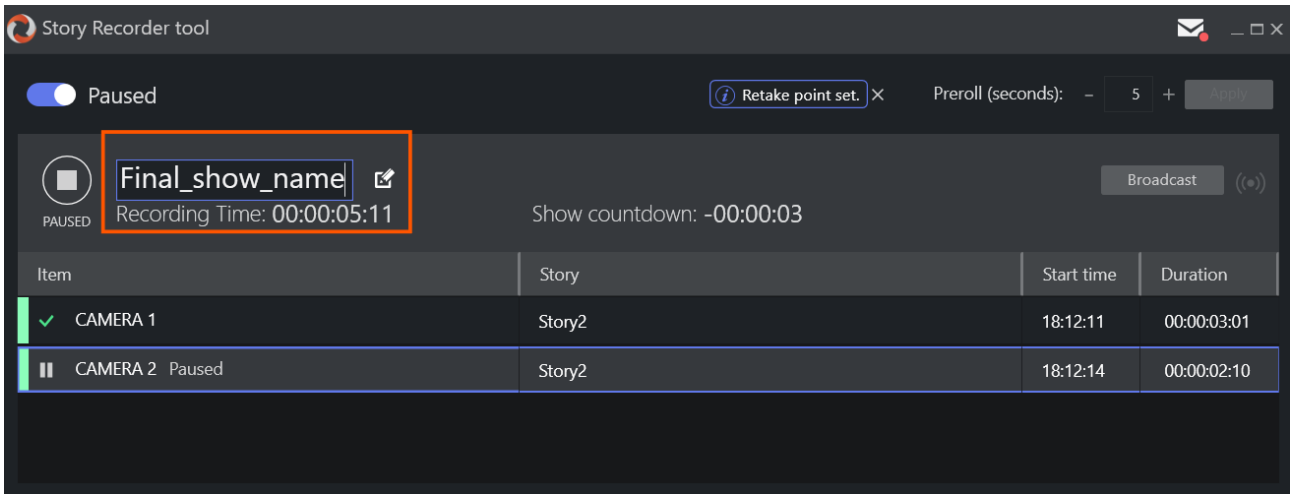
1. Enter a name in the NRCS as described above.
2. Load the rundown in the Viz Mosart UI.
3. Switch Story Recorder mode on. See Working with Story Recorder.
4. The name from the NRCS appears in the Story Recorder panel.

If the name does not appear, press **Ctrl+Shift+Alt** and double-click the first story item to inspect the underlying XML, and check whether the value arrived from the NRCS at all.

> The name is read when Story Recorder mode is switched on. Changing it in the NRCS during a running session has no effect on that session; the new name is picked up the next time Story Recorder mode is activated.

Operators can override the name:

The operator can type a name for the final clip directly in the Story Recorder panel. That name overwrites the one from the NRCS for that session.



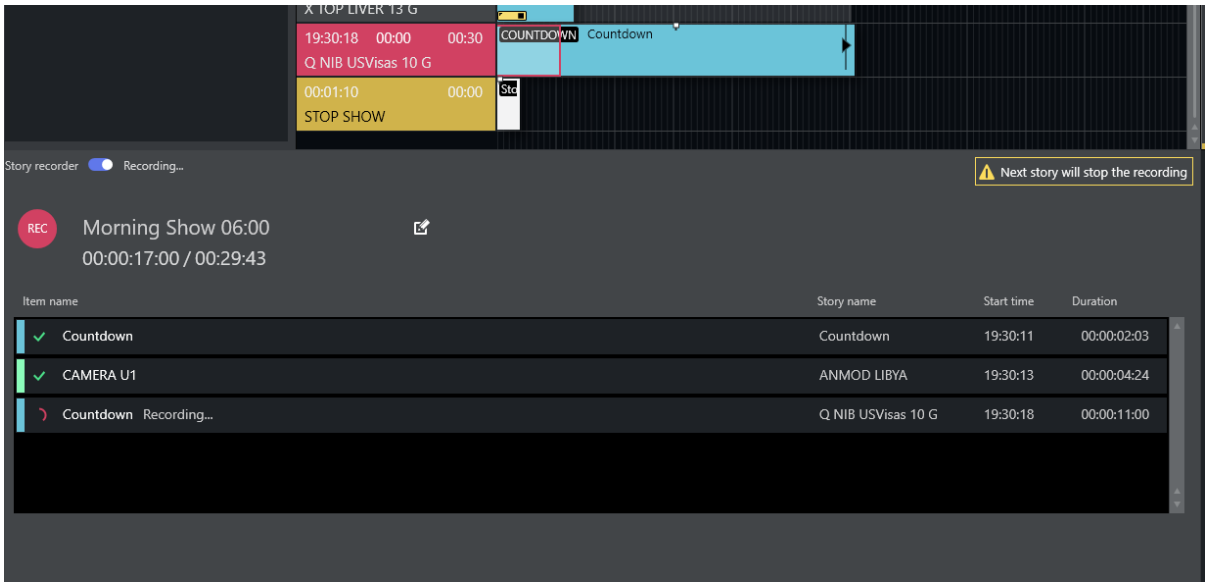
Set automatic pause at end of show

As an option, you can automatically pause the show and recording.

1. Create a dedicated template, typically a Break.
2. Define a `PAUSE_TIMELINE` control command configured on the template.

3. Insert this template at the end of the rundown.

In the Story Recorder session, when the story before this template is taken on air, a message warns the operator that the next story is going to pause the show and thus the recording too.



Assign keyboard shortcuts

The following Story Recorder actions exist only in the show clip production workflow. They are configured as **Control Command** shortcuts with the command `STORY_RECORDER` and an **Action** parameter:

Action	Description
PAUSE	Pauses the timeline while in Story Recorder mode.
RETAKE	Sets a retake point at the selected position.
BROADCAST_TIMELINE	Delivers the show recorded so far, the same as the **Deliver** button in the Story Recorder tool window (only available while paused).

A dedicated **Pause Timeline** shortcut is also available:

Shortcut name	Command	Description
Pause Timeline	`pause_timeline`	Pauses or resumes the Mosart timeline when in Story Recorder mode.

The shared `STORY_RECORDER` actions TOGGLE, ENABLE and DISABLE, which apply to all Story Recorder workflows, are described in **Keyboard Shortcuts** in Story Recorder Setup.

Clip naming pattern for EDL files and final show clips

Viz Mosart names EDL files and final show clips automatically.

First delivery: `<name>_<timestamp>.<ext>`

Subsequent deliveries of the same show: `<name>_v<n>_<timestamp>.<ext>`

Where:

- **name**: the Final Clip Show name, determined in this order of priority: value provided from the NRCS (`sr_final_clip_name`), or overridden by the operator in the Story Recorder panel, or the rundown name (default). See Configure automatic naming of the final clip.
- **timestamp**: automatically generated, format `YYMMDDHHmmss` .
- **ext**: depends on the output. `.tsedl` for Vantage EDL files, or the media extension configured on the transcoder for the final clip (for example, `.mxf`).
- `v<n>` : version counter, added from the second delivery onwards.

Example with Final Clip Show name `9AM news show - en` :

File	Description
`9AM news show - en_211011175050.tsedl`	EDL, first delivery
`9AM news show - en_211011175050.mxf`	Final clip, first delivery
`9AM news show - en_v2_211011175050.tsedl`	EDL, second delivery
`9AM news show - en_v2_211011175050.mxf`	Final clip, second delivery

Verification

1. Check that the **Notifications** in Mosart UI shows no genlock warning. *Unreliable genlock* or *Genlock is not enabled* means Viz Mosart fell back to the internal clock, and the clips will not be frame accurate. You can still verify Story Recorder general functionality without genlock and timecode setup though.
2. If you configured NRCS naming, load the rundown and confirm that the name from the NRCS appears in the Story Recorder UI. If it does not, press **Ctrl+Shift+Alt** and double click the first story item to inspect the underlying XML and check whether the value arrived.
3. Start the show and confirm that recordings appear on both the clean and dirty recording ports.
4. Press the **Pause Timeline** shortcut and confirm that the timeline pauses and the preroll template plays with a duration inside the **Min preroll duration** to **Max preroll duration** range.
5. Retake an item, then check the **Snippet clips folder** for a snippet clip that appears after the **Snippet generation delay**. Open it from the **Story Recorder** floating window in the Viz Mosart UI and confirm that the cut looks clean.
6. Pause the show and click **Deliver** in the Story Recorder tool window. A message such as *Delivered to Vantage* confirms what was sent.

Copyright © 2026 Vizrt

7. Switch Story Recorder mode off and confirm that the `.tsedl` file appears in both the **EDL local folder** and the **EDL watch folder**, named `<name>_<timestamp>.tsedl`.
8. Confirm in Vantage that the watch folder triggered clip assembly, and that the final show clip contains only the valid segments.

Operator instructions for switching on and running Story Recorder are in Working with Story Recorder.

Story Clip Production Setup

- Story Clip Production Setup
  - Choosing the clip output
  - Settings
  - NRCS Metadata for Story Clip Production
    - NRCS with built-in integration
    - NRCS via Newsroom Settings mapping (iNEWS, other systems)
  - Procedure
  - Verification

Story Clip Production Setup

This workflow records the stories that the NRCS marks for recording and produces one clip per story, either during a live broadcast or in an offline session. Two clip outputs are available: **TriCaster**, where TriCaster cuts and optionally publishes the clips, and **Mimir**, where Mimir renders and ingests the clips. This page covers what the two outputs share: the choice between them, the NRCS story metadata, and the setup sequence. The output-specific settings are on TriCaster Story Clip Setup and Mimir Story Clip Setup.

This page covers **step 19** and **step 22** of the Story Recorder Setup Checklist. Work through steps 1 to 18 first: they cover the license, the installation order, the frame accuracy setup, the show design, and the video server.

> Story Recorder needs frame accuracy in production. You can configure and test this output without genlock, but the marks will not be frame accurate. See steps 8 to 13 of the Setup Checklist.

Choosing the clip output

Component	TriCaster output	Mimir output
Recorder	TriCaster	Harmonic Spectrum
Transcoder/Renderer	TriCaster	Mimir
Online publishing	TriCaster publish presets (optional)	Not used in this workflow
Setup page	TriCaster Story Clip Setup	Mimir Story Clip Setup

> The recorder follows the renderer: TriCaster cuts story clips from its own recordings, and Mimir renders story clips from Harmonic Spectrum recordings. The video switcher is independent of this choice, so TriCaster can act as video server and recorder while a different switcher runs the show.

Both outputs work with any NRCS that can provide the required story metadata fields via the MOS protocol. For the validated combinations and the product versions, see **Supported Configurations** in Story Recorder Setup.

Settings

Open the Manus **Settings** menu and set **Enable Story Recorder** to `true`. The remaining Story Recorder settings sections then become available.

The **Clip Outputs** settings are specific to the output device. See TriCaster Story Clip Setup and Mimir Story Clip Setup.

> ⚠ Save while Story Recorder is off, and restart the **Mosart Logging Service** after changing the **Clip Outputs** devices. See step 25 of the Setup Checklist.

NRCS Metadata for Story Clip Production

Each story that should be recorded must carry metadata that tells Viz Mosart to produce a clip. The metadata is the same for both clip outputs.

NRCS with built-in integration

Some NRCSs (for example, *Saga*) provide built-in Story Recorder integration. They include a `<mosExternalMetadata>` block at the **story level** in the MOS message (`roStorySend`):

```
<mosExternalMetadata>
  <mosScope>STORY</mosScope>
  <mosSchema>http://www.vizrt.com/mosObj/storyPublisher</mosSchema>
  <mosPayload>
    <Record>true</Record>
    <Publish>false</Publish>
    <StoryClipId>394e83ee-55cc-48a7-a5f8-92db7cc36e10</StoryClipId>
    <StoryClipSlug>My Story Name_anchor_clip</StoryClipSlug>
  </mosPayload>
</mosExternalMetadata>
```

MOS Payload Element	Type	Purpose
`Record`	bool	*true*: Mark the story for clip production. *false*: Skip
`Publish`	bool	*true*: Also publish clip online (TriCaster only). *false*: Skip
`StoryClipId`	string	Clip identifier (for example, a Mimir anchor clip ID or a unique GUID)
`StoryClipSlug`	string	Human-readable clip description

The schema URI `http://www.vizrt.com/mosObj/storyPublisher` is what Viz Mosart matches on. The block can be present on every story; stories with `Record=false` are parsed but ignored by the publishing pipeline.

If another NRCS wants to support this integration, it needs to provide the same `<mosExternalMetadata>` structure with the `storyPublisher` schema.

NRCS via Newsroom Settings mapping (iNEWS, other systems)

For an NRCS that does not use the *storyPublisher* schema, the same metadata can be delivered via **MOS item variables**. The fields are then mapped to Mosart through the **Newsroom Settings** (`newsroomsettings.xml`) *Story External Metadata* configuration:

Mosart field name	Mosart action	Purpose
`sr_create_story_clip`	`item_variable`	Marks the story for clip production (`True` / `False`)
`sr_story_clip_id`	`item_variable`	Clip identifier (anchor clip ID or unique GUID)
`sr_story_clip_slug`	`item_variable`	Human-readable clip description
`sr_publish_story_clip`	`item_variable`	Marks for online publishing (`True` / `False` , TriCaster only)

The NRCS column names are mapped to the corresponding `sr_` action values. Both paths, the storyPublisher MOS schema and the Newsroom Settings mapping, produce identical results in Mosart. The following example shows how iNews columns (`var-20` , `var-21` , and so on) are mapped to Mosart `item_variable` actions with the corresponding `sr_` action values:

Newsroom Settings (c:\channeltemplates\newsroomsettings.xml)

File Edit **Story External Metadata**

mostagname	Mosart Action	Format Type	mos_value	action_value
col04	item_variable	String		lang
var-007	item_variable	String		sr_final_clip_name
channel	directtake_pre_story	String	pre	101
col01	directtake_post_story	String	post	102
var-1	template_type	String		
var-2	template_variant	String		
var-3	template_transition	String	test	Effekt|12
var-4	item_variable	String		clip_hirespath
var-5	story_variable	String		xpoint1
var-6	story_variable	String		xpoint2
var-7	item_variable	String		output_preview
col03	item_duration	frames		
var-8	story_endphrase	String		
OutCue	story_endphrase	String		
OutCue2	story_endphrase	String		
ClipDescription1	field_value_1	String	test	clip_description=something
ClipHirespath	field_value_2	String		clip_hirespath
mTransition	template_transition	String	test	EFFEKT 2
var-21	item_variable	String		sr_story_clip_id
var-20	item_variable	String		sr_create_story_clip
var-22	item_variable	String		sr_publish_story_clip
var-23	item_variable	String		sr_story_clip_slug

Procedure

1. In the Manus **Settings** menu, enable **Enable Story Recorder**.
2. Configure the clip output on the device page: TriCaster Story Clip Setup or Mimir Story Clip Setup.
3. Configure the NRCS story metadata as described above.
4. Save the settings while Story Recorder is off. If you changed the **Clip Outputs** devices, restart the **Mosart Logging Service** on the Mosart Server machine.

Verification

Each device page ends with a numbered verification flow for its output. See **Verification** in TriCaster Story Clip Setup or Mimir Story Clip Setup.

Operator instructions for switching on and running Story Recorder are in Working with Story Recorder.

Mimir Story Clip Setup

- Mimir Story Clip Setup
  - What you need
  - Settings
    - Clip Outputs > Mimir
    - Transcription and request templates
    - Related settings
  - Procedure
    - Configure the Mimir output
    - Configure the NRCS story metadata
  - Re-recording a story
  - Verification

Mimir Story Clip Setup

This workflow records the stories that the NRCS marks for recording and produces one clip per story in Mimir. Viz Mosart records the program output on the video server, then asks Mimir to render each story segment out of the growing recording and ingest it as a clip. Operators can run it during a live broadcast or in an offline session.

This page covers the Mimir half of **step 19** of the Story Recorder Setup Checklist. Work through steps 1 to 18 first. The setup shared by both story clip outputs, including the NRCS story metadata, is in Story Clip Production Setup.

> Story Recorder needs frame accuracy in production. You can configure and test this output without genlock, but the marks will not be frame accurate. See steps 8 to 13 of the Setup Checklist.

What you need

On top of the general Story Recorder setup, this output needs:

- A **Generic REST device** configured in Viz Mosart. This device handles the API communication with Mimir. See more details below.
- A render job preset and a Kelda configuration set up in Mimir **System Settings**.
- A transcription configuration in the Mimir tenant settings that matches the language code and transcription service you configure here.

For the validated combination of NRCS, recorder and renderer, and the product versions, see **Supported Configurations** in Story Recorder Setup.

Settings

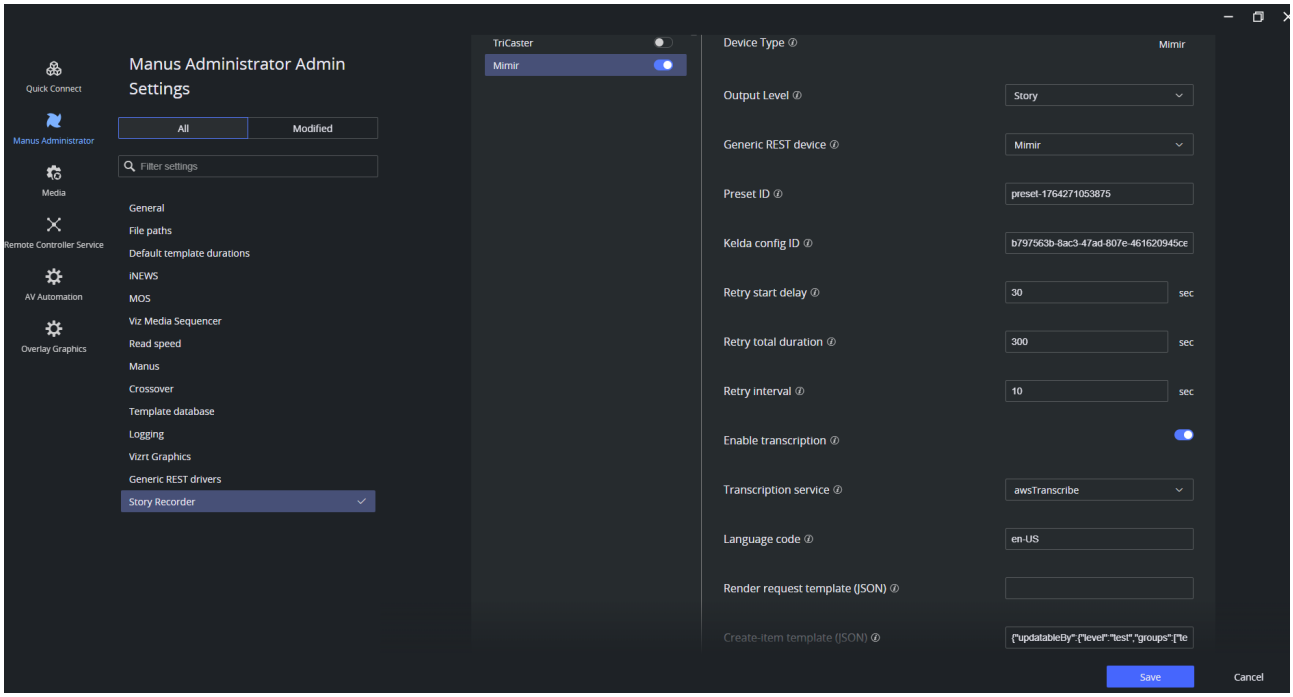
Open the Manus **Settings** menu and set **Enable Story Recorder** to `true`. The remaining Story Recorder settings sections then become available.

> Save while Story Recorder is off, and restart the **Mosart Logging Service** after changing the **Clip Outputs** devices, including this Mimir output. See step 25 of the Setup Checklist.

Clip Outputs > Mimir

Setting	Default	Description
Output Level	Story	Read-only. Determines the granularity at which clips are produced. Currently, only *story-level* clips are supported with Mimir. At story-level, one clip is generated per rundown story.
Generic REST device	*(empty)*	The Generic REST device configured for Mimir. This device handles the API communication with Mimir.
Preset ID	*(empty)*	Identifier of the render job preset configured in Mimir **System Settings**. Determines the render profile used when producing the story clip.
Kelda config ID	*(empty)*	The Kelda configuration ID, set up in Mimir **System Settings**. Kelda is Mimir's ingest orchestration component, this ID informs Mimir which ingest configuration to use when receiving the recorded media from the video server.
Retry start delay	`30` sec	Delay before Story Recorder starts checking ingest status and retrying, after the publish command is sent to Mimir. Allows time for the initial render/ingest to start.
Retry total duration	`300` sec	Total time to spend on ingest checking and render retries before giving up. If all retries are exhausted, the publish is reported as failed.
Retry interval	`10` sec	Time between consecutive ingest checks and render retries.

Copyright © 2026 Vizrt



Transcription and request templates

Setting	Default	Description
Enable transcription	`true`	Request a transcript for the rendered story clip.
Transcription service	`awsTranscribe`	Transcription service provider Mimir uses. The (language code, service) pair must match a transcription configuration in the Mimir tenant settings. For a provider not listed, set it via the render template.
Language code	`en-US`	BCP-47 language code for transcription (e.g. en-US). The (language code, service) pair must match a transcription configuration in the Mimir tenant settings.
Render request template (JSON)	*(empty)*	JSON merged into the Mimir render request: analyticConfig, delegate, visibleTo, output profile.
Create-item template (JSON)	*(empty)*	JSON merged into the Mimir create-item (placeholder) request: item type, transcription, access control, folders.
Create-version template (JSON)	*(empty)*	JSON merged into the Mimir create-version request: archive linking, media copy, name.

Values available for **Transcription service**: `awsTranscribe`, `googleSpeechToText`, `ibmSpeechToText`, `msSpeechToText`, `zoomSpeechToText`, `speechmaticsSpeechToText`,

`trintSpeechToText`, `amberscriptSpeechToText`, `recappSpeechToText`, `recapp2SpeechToText`, `srtImport`.

> ⚠ A language code and transcription service pair that does not match a transcription configuration in the Mimir tenant settings fails the render.

> ℹ Invalid JSON in `Render request template (JSON)`, `Create-item template (JSON)`, or `Create-version template (JSON)` logs a warning and falls back to the defaults. The call itself does not fail.

Related settings

These settings are shared by all Story Recorder workflows and live on Story Recorder Setup:

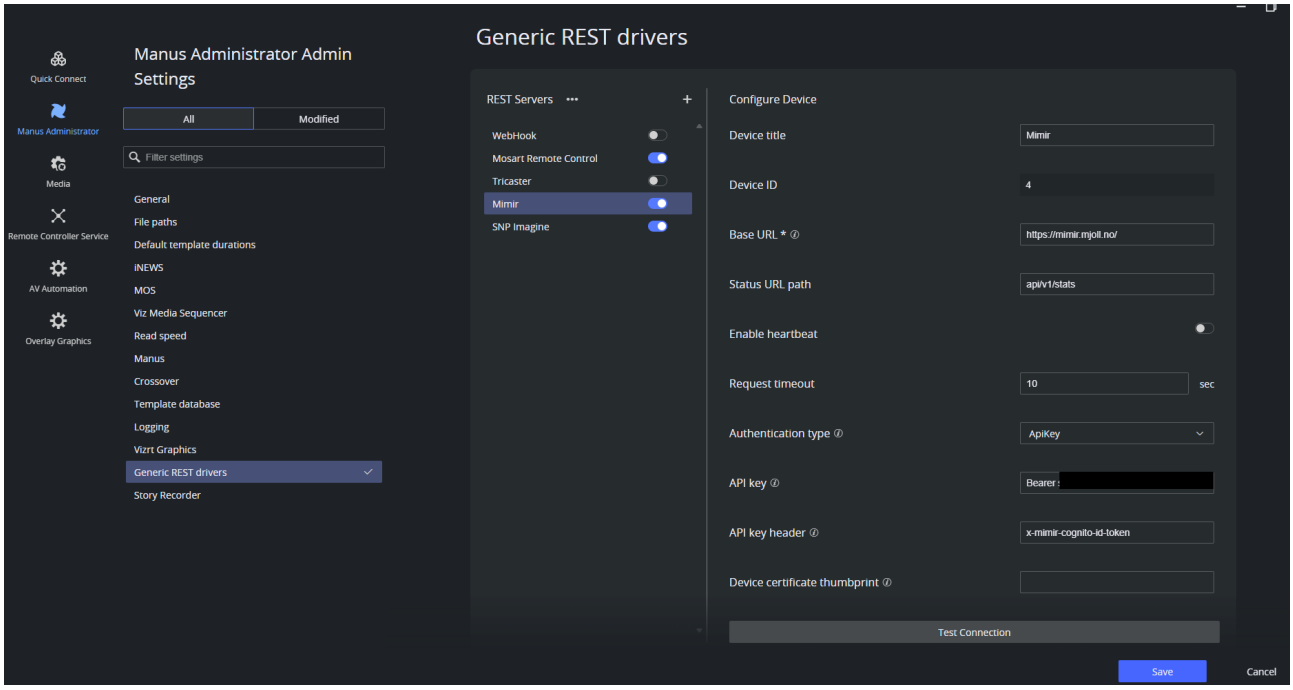
- **Recordings Housekeeping** and **Related General Setting**: the recording clip name prefix, the retention days, and **Use local time**.
- **Keyboard Shortcuts** and **Clip naming pattern of temporary recordings**.

Procedure

Configure the Mimir output

1. In the Manus **Settings** menu, configure Mimir as a Generic REST device.
2. In the Manus **Settings** menu, enable **Enable Story Recorder**.
3. Under **Clip Outputs**, open the **Mimir** section.
4. Select the **Generic REST device** configured for Mimir.
5. Enter the **Preset ID** of the render job preset from Mimir **System Settings**.
6. Enter the **Kelda config ID** from Mimir **System Settings**.
7. Leave **Retry start delay** (`30` sec), **Retry total duration** (`300` sec), and **Retry interval** (`10` sec) at their defaults unless your Mimir tenant needs longer.
8. Set **Enable transcription**, **Transcription service**, and **Language code**. Confirm that the language code and service pair matches a transcription configuration in the Mimir tenant settings.
9. Add JSON to **Render request template (JSON)**, **Create-item template (JSON)**, or **Create-version template (JSON)** only if you need to override the defaults.
10. Save the settings and restart the **Mosart Logging Service** on the Mosart Server machine.

Copyright © 2026 Vizrt



Configure the NRCS story metadata

Each story that should be recorded must carry metadata that tells Viz Mosart to produce a clip. This configuration is shared by all story clip outputs, and is documented in **NRCS Metadata for Story Clip Production** in Story Clip Production Setup. Two paths deliver the same result: an NRCS with built-in integration (for example, Saga) sends a `storyPublisher` metadata block, while any other NRCS (for example, iNEWS) delivers the same fields through Newsroom Settings field mapping. The Newsroom Settings mapping for iNEWS looks like this:

Newsroom Settings (c:\channeltemplates\newsroomsettings.xml)

File Edit **Story External Metadata**

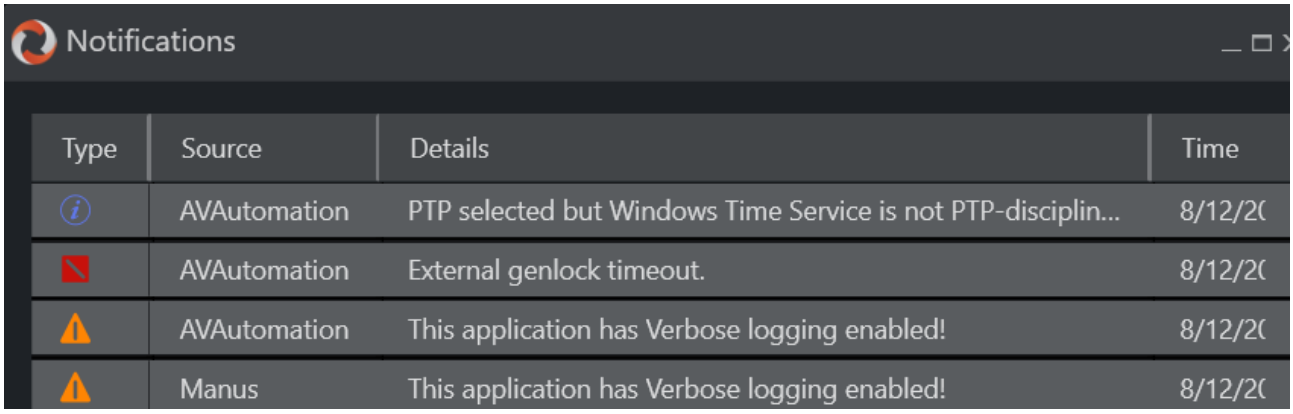
mostagname	Mosart Action	Format Type	mos_value	action_value
col04	item_variable	String		lang
var-007	item_variable	String		sr_final_clip_name
channel	directtake_pre_story	String	pre	101
col01	directtake_post_story	String	post	102
var-1	template_type	String		
var-2	template_variant	String		
var-3	template_transition	String	test	Effekt|12
var-4	item_variable	String		clip_hirespath
var-5	story_variable	String		xpoint1
var-6	story_variable	String		xpoint2
var-7	item_variable	String		output_preview
col03	item_duration	frames		
var-8	story_endphrase	String		
OutCue	story_endphrase	String		
OutCue2	story_endphrase	String		
ClipDescription1	field_value_1	String	test	clip_description=something
ClipHirespath	field_value_2	String		clip_hirespath
mTransition	template_transition	String	test	EFFEKT 2
var-21	item_variable	String		sr_story_clip_id
var-20	item_variable	String		sr_create_story_clip
var-22	item_variable	String		sr_publish_story_clip
var-23	item_variable	String		sr_story_clip_slug

Re-recording a story

When a story is re-recorded, Mimir creates a new version under the same anchor clip. Mimir (not Viz Mosart) notifies the NRCS of the updated anchor clip.

Verification

1. Load a rundown that has at least one story marked for recording, and check that the story items appear in the **Story Recorder** panel.
2. Check that the **Notifications** in Mosart UI shows no genlock warning. *Unreliable genlock* or *Genlock is not enabled* means Viz Mosart fell back to the internal clock, and the clips will not be frame accurate. You can still verify Story Recorder general functionality without genlock and timecode setup though.



3. Switch Story Recorder on in the Viz Mosart UI. The session must start without errors. Otherwise, check Story Recorder Troubleshooting.
4. Take the first story on air and confirm that an intermediate recording appears on the video server, named `<prefix>_<counter>_<timestamp>.<ext>` (for example, `SRVizrt_1_211001153956.mxf`).
5. When the story ends, confirm in Mimir that the story clip is created with the expected render profile.
6. If **Enable transcription** is on, check that Mimir produces the transcript with the configured service and language code.
7. Re-record the same story and confirm that Mimir adds a new version under the same anchor clip.

Operator instructions for switching on and running Story Recorder are in Working with Story Recorder.

TriCaster Story Clip Setup

- What you need
- Settings
- Procedure
- Verification

TriCaster Story Clip Setup

This workflow records the stories that the NRCS marks for recording and produces one clip per story with TriCaster. TriCaster acts as recorder, transcoder, and online publisher: Viz Mosart records the program output, then sends an export job that cuts the story clip out of the growing recording. When a story is also marked for publishing, TriCaster pushes the clip to the online platforms in its publish presets.

This page covers the TriCaster half of **step 19** of the Story Recorder Setup Checklist. Work through steps 1 to 18 first, including **step 16**, which prepares TriCaster itself. The setup shared by both story clip outputs, including the NRCS story metadata, is in Story Clip Production Setup.

> **Info**: Story Recorder needs frame accuracy in production. You can configure and test this output without genlock, but the marks will not be frame accurate. See steps 8 to 13 of the Setup Checklist.

What you need

On top of the general Story Recorder setup, this output needs:

- Export presets configured in TriCaster. Story Recorder can name them explicitly, or fall back to the presets currently enabled in TriCaster.
- Publish presets configured in TriCaster, if you want online publishing.
- HTTP credentials for the TriCaster REST API. Viz Mosart uses them to request Start of Media (the timecode of the first frame) for the render job command.

For the validated combination of NRCS, recorder and transcoder, and the product versions, see **Supported Configurations** in Story Recorder Setup.

Settings

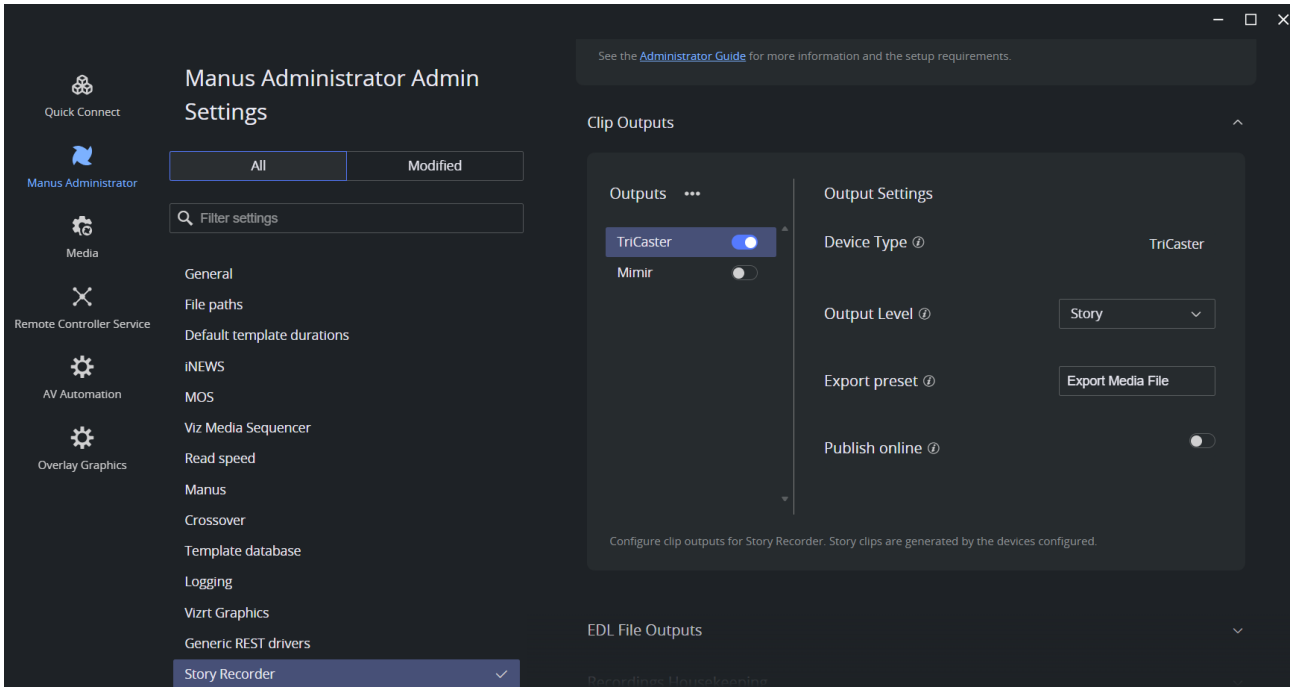
Open the Manus **Settings** menu and set **Enable Story Recorder** to `true`. The remaining Story Recorder settings sections then become available.

> **Note**: Save while Story Recorder is off, and restart the **Mosart Logging Service** after changing the **Clip Outputs** devices, including this TriCaster output. See step 25 of the Setup Checklist.

Clip Outputs > TriCaster

Setting	Defaul t	Description
Output Level	Story	Read-only. Determines the granularity at which clips are produced. At *Story* level, one clip is generated per story in the rundown. Currently, only story-level output is supported for the workflow with TriCaster.
Export preset	*(empty)*	TriCaster export preset name(s). Multiple presets can be separated by a vertical bar. If empty, the export presets currently enabled in TriCaster will be used.
Publish online	`false`	When enabled, clips will also be published to the online platforms as specified in the TriCaster's publish presets.
Publish preset	*(empty)*	TriCaster publish preset name(s) for online publishing. Multiple presets can be separated by a vertical bar. Visible only when **Publish online** is enabled. If empty, the publish presets currently configured in TriCaster will be used.

Preset value examples: a single preset is `Export Media File`, and two presets are `Preset1|Preset2`. The same syntax applies to **Export preset** and **Publish preset**.



Related settings

These settings are shared by all Story Recorder workflows and live on Story Recorder Setup:

- **Preparing TriCaster**: the TriCaster session frame rate, genlock and timecode, the recording folder, and the `HttpUsername`, `HttpPassword` and `AddExportJobDelayMs` connection string properties.
- **Recordings Housekeeping** and **Related General Setting**: the recording clip name prefix, the retention days, and **Use local time**.
- **Keyboard Shortcuts** and **Clip naming pattern of temporary recordings**.

> Recorded clips can be *very large* files, easily filling the disk space.

Procedure

Configure the TriCaster clip output

1. In the Manus **Settings** menu, enable **Enable Story Recorder**.
2. Under **Clip Outputs**, open the **TriCaster** section.
3. Enter the **Export preset** name, or several names separated by a vertical bar. Leave it empty to use the export presets currently enabled in TriCaster.
4. Enable **Publish online** if clips should also go to the online platforms.
5. With **Publish online** enabled, enter the **Publish preset** name(s), separated by a vertical bar for several. Leave it empty to use the publish presets currently configured in TriCaster.
6. Save the settings and restart the **Mosart Logging Service** on the Mosart Server machine.

> **Info**: The Quick Connect feature in Manus Settings configures the video server setup, enables the recorder for Story Recorder, and writes the `HttpUsername` and `HttpPassword` properties. To configure the ports manually, or to check what Quick Connect produced, see **Configuring the Video Server** in Story Recorder Setup.

Configure the NRCS story metadata

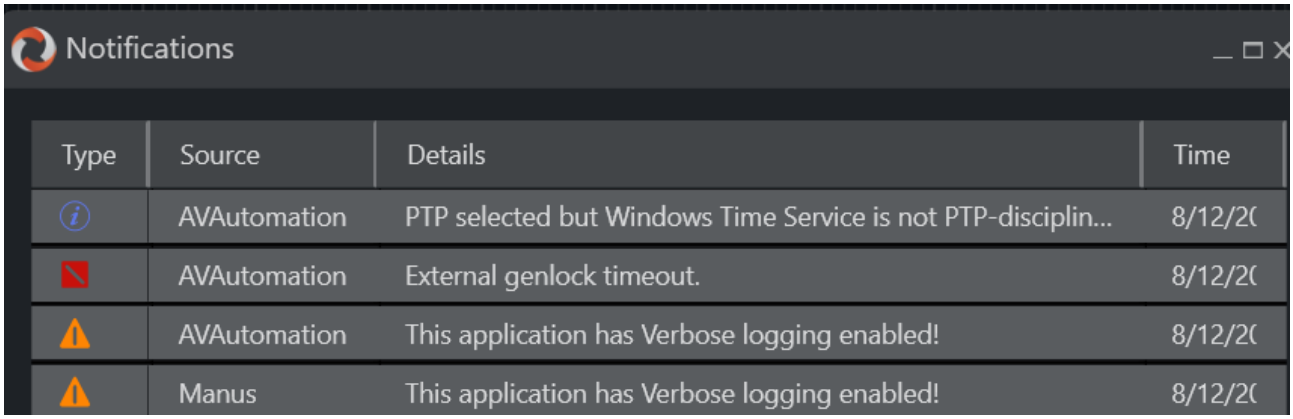
Story metadata configuration is shared by all story clip outputs: each story that should be recorded must carry metadata that tells Viz Mosart to produce a clip, delivered either by an NRCS with built-in integration or by Newsroom Settings field mapping. The `<mosExternalMetadata>` sample, the MOS payload elements and the Newsroom Settings field names are documented in **NRCS Metadata for Story Clip Production** in Story Clip Production Setup.

Online publishing is specific to TriCaster and is controlled by the `Publish` payload element, or by the `sr_publish_story_clip` mapped field.

Verification

1. Load a rundown that has at least one story marked for recording, and check that the story items appear in the **Story Recorder** panel.
2. Check that the **Notifications** in Mosart UI shows no genlock warning. *Unreliable genlock* or *Genlock is not enabled* means Viz Mosart fell back to the internal clock, and the clips will not be frame accurate. You can still verify Story Recorder general functionality without genlock and timecode setup though.

Copyright © 2026 Vizrt



3. Switch Story Recorder on in the Viz Mosart UI. The session must start without errors. Otherwise, check Story Recorder Troubleshooting.
4. Take the first story on air and confirm that an intermediate recording appears on the video server, with the `mov` extension and a name such as `SRVizrt_1_211001153956.mov`.
5. When the story ends, confirm that TriCaster runs the export job. Allow for the `AddExportJobDelayMs` delay (5 seconds by default) before the export starts.
6. If **Publish online** is enabled, confirm on the target online platform that TriCaster published the clip with the configured publish preset.

Operator instructions for switching on and running Story Recorder are in Working with Story Recorder.

Mosart EDL Setup

Mosart EDL Setup

This output makes Viz Mosart write the edit decisions from a Story Recorder session to disk as JSON files. No external transcoder or renderer takes part: Viz Mosart records the program output on the video server, logs every on-air event with a frame-accurate timecode, and writes the resulting in and out points to a folder you configure.

Enable it *alongside* one of the other Story Recorder outputs. The Mosart EDL produces no clips of its own, so it never replaces the output that produces your media. Its main purpose is **troubleshooting**: it gives you, and Vizrt support, a readable record of what a session recorded. The same files can also feed a third-party system that cuts or catalogues the recording, or a converter that turns the timings into another EDL format, but that is not a finished workflow yet. If you are considering it, contact Vizrt to learn how the output works and to develop it further together.

This page covers the Mosart EDL half of **step 19** of the Story Recorder Setup Checklist. Work through steps 1 to 18 first: they cover the license, the installation order, the frame accuracy setup, the show design, and the video server.

> Story Recorder needs frame accuracy in production. You can configure and test this output without genlock, but the timings in the files will not be frame accurate: the in and out points no longer match the recording frame for frame. See steps 8 to 13 of the Setup Checklist.

What this workflow is for

The intent. Viz Mosart writes the Edit Decision List to disk as JSON. It produces timing information only: it renders no clip and it moves no media. The files tell you, item by item, which recording each part of the show is in and which frames it covers.

What you get. JSON files in a folder you configure. Each file names the recorded media, the point where that recording starts, and the frame range of every item that was recorded.

When to use it: troubleshooting. This is what the output is for today. Use it to check what a session actually recorded, or when Vizrt asks for the files while investigating a timing question. Enable it at the same time as the Mimir, TriCaster or Vantage output. It runs independently of them and does not interfere with them.

Third-party integrations. The files also carry what another system would need to cut, stitch or catalogue the recording itself. The most likely case is a converter of your own: read the JSON and write out the EDL or playlist format that your transcoder, editing system or MAM expects. This is not a finished workflow, and the format is still developing. Contact Vizrt before you build on it, so you learn how the output behaves today and your requirements can shape where it goes next.

Scope and limits

> **This output is still under development, and the format is not final.** It carries no version number and no published schema, and it can change from one Viz Mosart release to the next. Use it for troubleshooting. Before any third-party system depends on these files, contact Vizrt, so that what you need is taken into account as the output develops.

- Viz Mosart writes the files and does nothing more. It does not send them anywhere, it does not tell any system that a new file has appeared, and it does not delete them later. Whatever consumes the files has to look in the folder itself.

- The files describe timings, not media. No clip is produced and nothing is rendered. The recording stays on the video server.
- One file per item, per story or per show, depending on the output levels you select. The files go into `Item\`, `Story\` and `Show\` sub-folders of the folder you configure.
- Show level files are written only when the operator presses **Deliver** in the Story Recorder tool window. A session that runs without a **Deliver** press produces no Show level file.
- A pause and retake is reflected in the files: the discarded part of the show is removed, and the item recorded again carries a higher **take** number. Cut from the item entries rather than the story ones, because a story's own start and end timecodes still span the discarded material and the pause. And where a story was written to disk before the retake, its earlier file stays behind, so the newest file for a story is the valid one.
- Graphics are only partly covered. Only overlay graphics appear, and their timings come from the clock rather than from genlock, so they are **not** frame accurate the way the item timings are. Fullscreen graphics appear as ordinary items, with nothing marking them as graphics. Graphics channel, type and description are not included.
- Several things are not in the files at all: audio, transitions and transition types, item types, switcher crosspoints, accessories, control commands, and clip in and out points. A converter can only pass on what the files contain, so anything in this list has to come from elsewhere.
- JSON only. This output never produces the Vantage `.tsedl` file. That format belongs to the show clip production workflow.

What you need

On top of the general Story Recorder setup, this output needs:

Component	Requirement
Output path	A local path or a network share for the JSON files.
Extra software	None. Viz Coder is not used, because this workflow produces no snippet clips.
Other Story Recorder output	Recommended. This output produces no media of its own, so enable it next to the Mimir, TriCaster or Vantage output that produces your clips.

For the validated combination of NRCS, recorder and transcoder, and the product versions, see **Supported Configurations** in Story Recorder Setup.

Settings

Open the Manus **Settings** menu and set **Enable Story Recorder** to `true`. The remaining Story Recorder settings sections then become available.

> ⚠ Save while Story Recorder is off, and restart the **Mosart Logging Service** after changing the **Clip Outputs** devices. See step 25 of the Setup Checklist.

EDL File Outputs > Mosart

Produces JSON-format EDL files for local inspection of a recorded session.

Setting	Default	Description
Output Level	Story	Mosart supports **Item**, **Story**, and **Show** level EDL output. Select one or more levels; a file is written at each level you select, into the matching sub-folder. If you clear the selection, no files are written.
EDL local folder	`C:\MMLogs\EDL`	Folder where JSON EDL files are written. Files are organized into `Item/`, `Story/`, and `Show/` sub-folders based on the selected output level. Can be a local path or a network share.

Output levels

The levels you select determine which files Viz Mosart writes, and which sub-folder they go to.

Level	Subfolder	What it produces
Item	`Item/`	One EDL per rundown item recorded during the session. The finest granularity available.
Story	`Story/`	One EDL per story in the rundown.
Show	`Show/`	One EDL per show, covering the whole session.

Selecting several levels writes a file at each of them, from the same session. An empty selection writes nothing.

Show level. Show level files are written only when the operator presses **Deliver** in the Story Recorder tool window. Disabling Story Recorder does not write a Show level file, so a session that runs without a **Deliver** press produces no Show level output, and the file cannot be created after the session ends.

Reading the frame ranges. The out point is the first frame *after* an item, not its last frame. The duration of an item is therefore its out point minus its in point, and items follow each other with no gap and no overlap.

Related settings

These settings are shared by all Story Recorder workflows and live on Story Recorder Setup:

- **Recordings Housekeeping** and **Related General Setting**: the recording clip name prefix, the retention days, and **Use local time**.
- **Keyboard Shortcuts** and **Clip naming pattern of temporary recordings**.

Viz Mosart does not clean up the **EDL local folder**. The JSON files are small, but they need occasional purging. Recorded clips on the video server are a different matter: they can be *very large* files, easily filling the disk space.

Procedure

Configure the video server recorder ports

See **Configuring the Video Server** in Story Recorder Setup. With TriCaster as recorder, also work through **Preparing TriCaster** on the same page.

Configure the Mosart EDL output

1. In the Manus **Settings** menu, enable **Enable Story Recorder**.
2. Under **EDL File Outputs**, open the **Mosart** section.
3. In **Output Level**, select the levels you need: **Item**, **Story** or **Show**. Selecting several writes a file at each of them. If you leave the selection empty, no files are written.
4. Set **EDL local folder**. Keep `C:\MMLogs\EDL` or choose another local path or network share. Grant the *Mosart Logging Service* account read and write permission on it. Viz Mosart creates a sub-folder for each level you selected.
5. Save the settings while Story Recorder is off. They take effect when the next session is enabled.

Verification

1. Switch Story Recorder on in the Viz Mosart UI. The session must start without errors. Otherwise, check Story Recorder Troubleshooting.
2. Load a rundown and check that the story items appear in the **Story Recorder** panel.
3. Check that the **Story Recorder** panel shows no genlock warning. *Unreliable genlock* or *Genlock is not enabled* means Viz Mosart fell back to the internal clock. The session still runs and the files are still written, but their timings are not frame accurate.
4. Check that the **EDL local folder** contains a sub-folder for each output level you selected, and none for the levels you did not select.
5. Run a short session, then confirm that JSON EDL files land in the sub-folder of every level you selected: one per item in `Item/`, one per story in `Story/`, one per show in `Show/`. At Show level, the operator must press **Deliver** when in Pause mode for a file to appear. A message such as *Delivered to Mosart EDL (6 items, 4 stories, 1 show)* confirms what was written.
6. If your setup uses pause and retake, retake an item and then confirm that the story file lists that item once, with its **take** number counted up, and not twice.
7. Open one of the JSON files and confirm that the timecodes match the recordings on the video server. A systematic offset points to a mismatch between **Use local time** and the AV Automation setting **External Clock Is Local Time**.

Operator instructions for switching on and running Story Recorder are in Working with Story Recorder.

12.4.5 Working with Story Recorder

- Managing a Story Recorder Session
  - Enable/disable Story Recorder mode
  - Story Recorder panel
- Show Clip Production with Pause and Retake
  - Recording a Show as a Single Clip
  - Pause-Fix-Retake Mode
  - Broadcasting an Alternative Version of the Show
- Story Clip Production and Publishing
  - Stories Marked for Recording
  - Clip Production Workflows
  - TriCaster Clip Export
  - Mimir Clip Production
  - Using Recorded Story Clips
- Timing Details

This section describes how to operate Story Recorder (SR) during a production, covering two workflows: story clip production, and show clip production with pause and retake.

For an overview of Story Recorder and its workflows, see section Story Recorder.

> Story Recorder can also write a Mosart EDL file. An administrator configures that output, and it produces files for inspection rather than anything you drive from the Story Recorder panel. See Mosart EDL Setup.

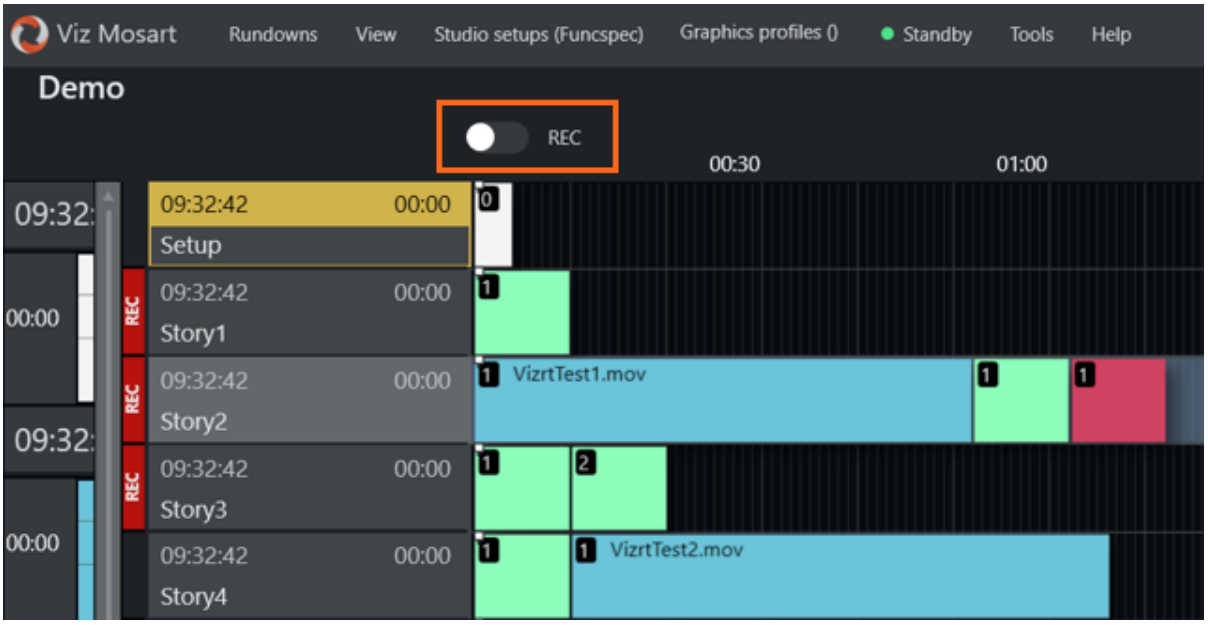
Managing a Story Recorder Session

Enable/disable Story Recorder mode

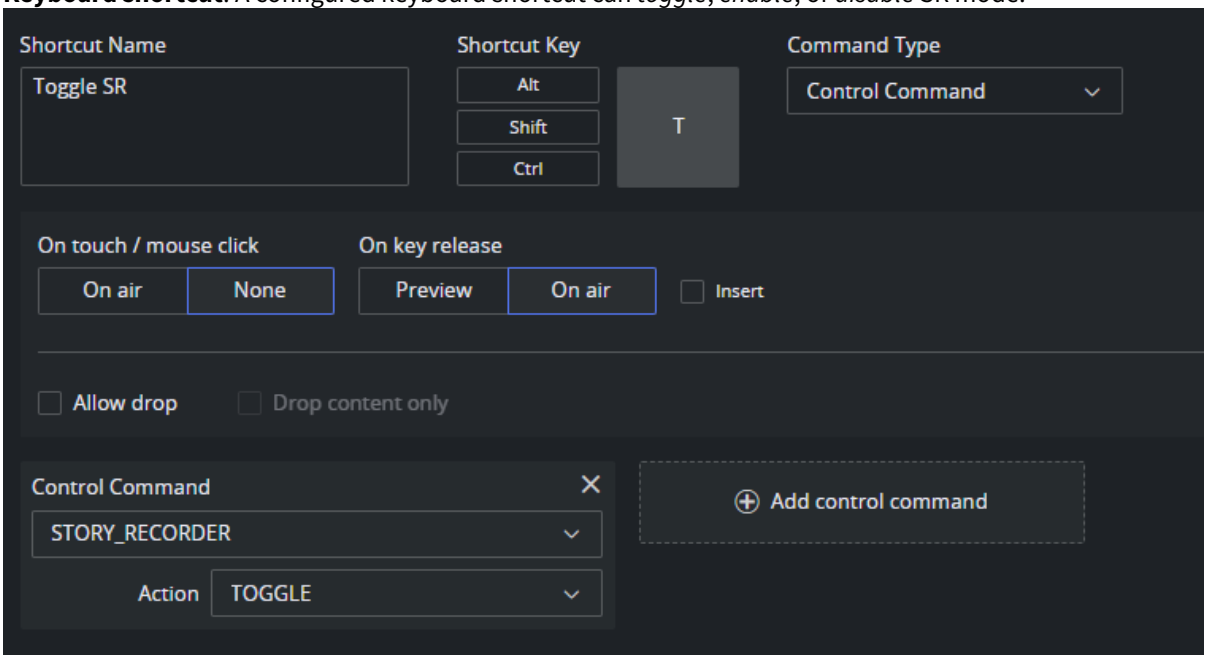
You can enable or disable Story Recorder mode using any of these methods:

1. **Story Recorder panel toggle**: in the Viz Mosart UI, the On/Off switch at the top of the menu bar selection **View > Floating windows > Story Recorder tool**.

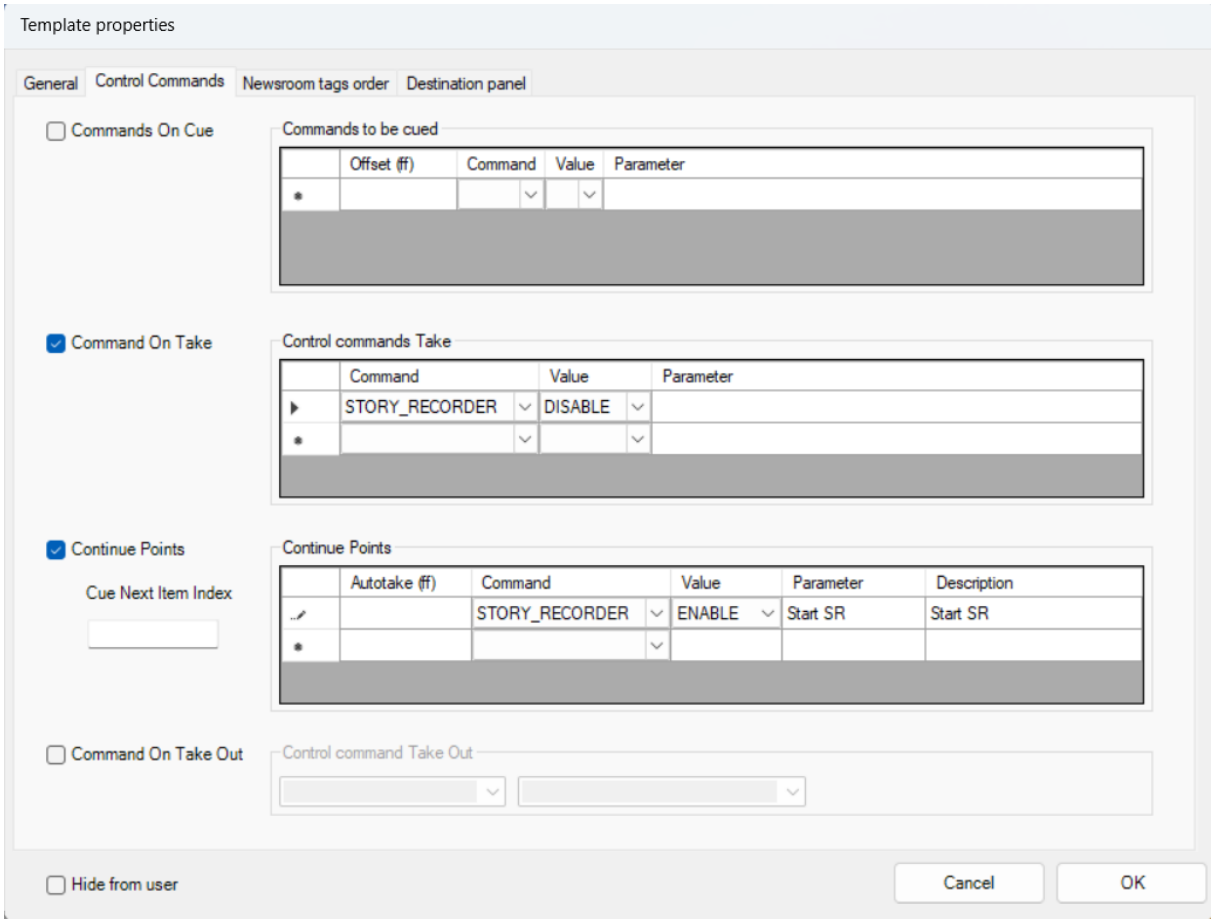
2. **Timeline toolbar toggle**: When the rundown contains stories that have been marked in the NRCS for recording, an SR toggle button (**REC**) appears directly in the Viz Mosart UI timeline toolbar:



3. **Keyboard shortcut**: A configured keyboard shortcut can *toggle*, *enable*, or *disable* SR mode.

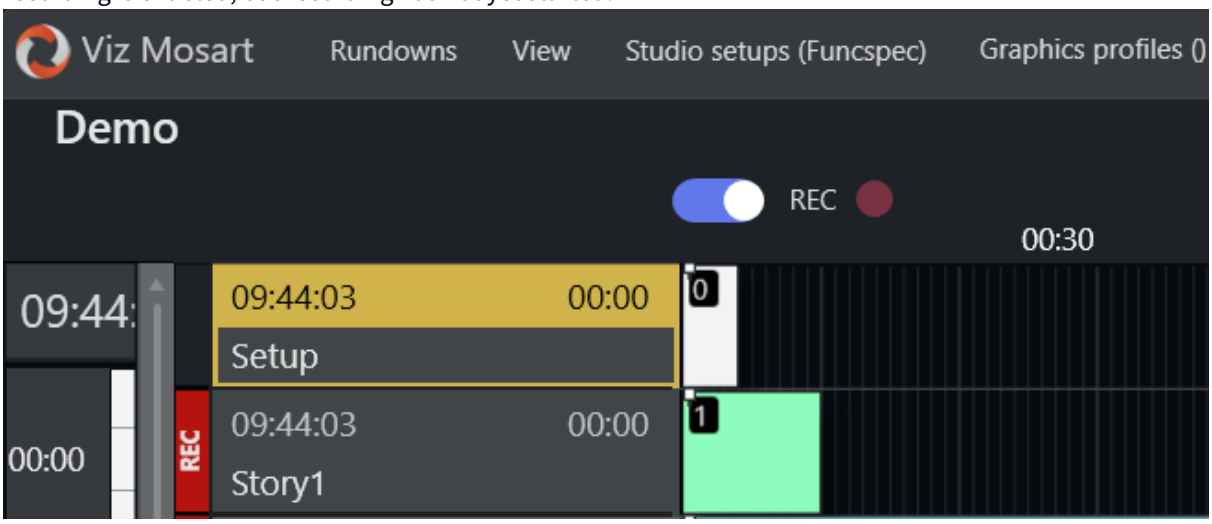


4. **Template control command**: the `STORY_RECORDER` control command on a template can automatically *toggle*, *enable*, or *disable* SR mode when the template is taken.



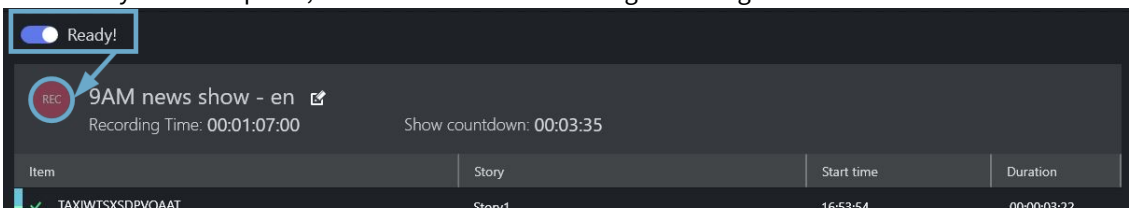
All four methods produce the same result. The current recording *status* is shown in both the Timeline toolbar and the Story Recorder panel:

- ***Pale red* REC indicator**: Story Recorder mode in the toolbar above the Timeline indicates that story recording is enabled, but recording has not yet started.

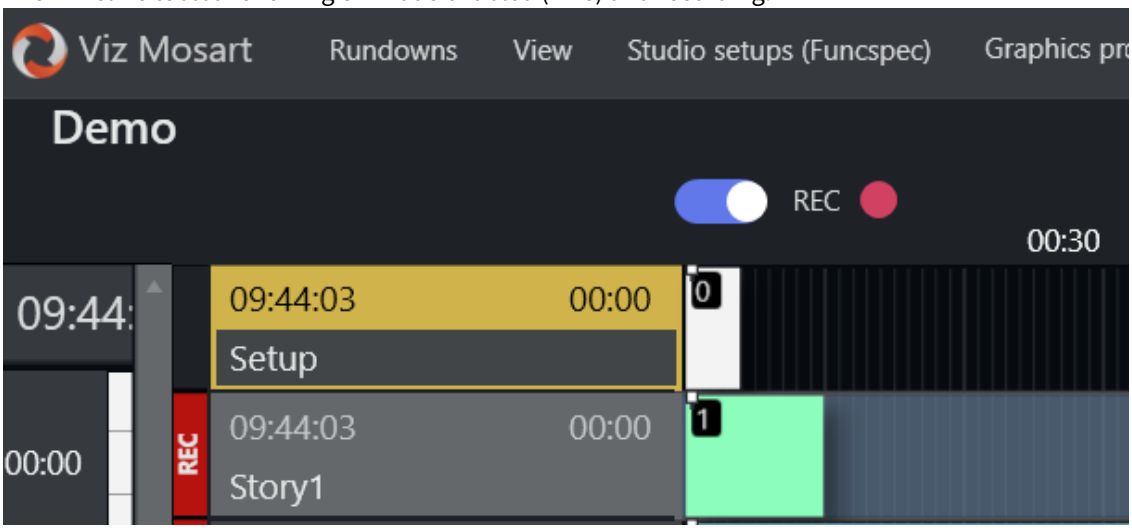


- ***Bright red* REC blinking indicator**: Story Recorder is actively recording.

- In the Story Recorder panel, the REC button blinks during recording.



- The Timeline toolbar showing SR mode enabled (**REC**) and recording.



Info:

- The Story Recorder panel is required for the *show clip production* workflow (pause, retake, broadcast).
- For the *story clip production* workflow, the panel is not necessary.
- When there are stories tagged for recording, the SR toggle and status are shown directly in the Viz Mosart timeline, together with a red **REC** indicator alongside the tagged stories (for example, *Story1* above).

Story Recorder panel

The Story Recorder panel provides a real-time view of the EDL as it is being built during a recording session. It displays each story item and story as they are recorded, together with start times, durations, and recording status. The panel is primarily used with workflows that produce an EDL output (Vantage or Mosart format), where the operator can pause the timeline, select retake points, and resume recording. At the end of the show, the completed EDL can be broadcast to the Vantage transcoder for stitching into a final clip, or exported as a Mosart EDL file.

To display Story Recorder Panel, follow these steps:

1. With Viz Mosart running, from the menu bar select **View > Floating windows > Story Recorder tool**.
2. Position the Story Recorder window on your workspace.
   (You can also place this window on a second screen.)
3. (Optional) Save this workspace setup for future use:
   View > Save as new workspace and enter a unique name.

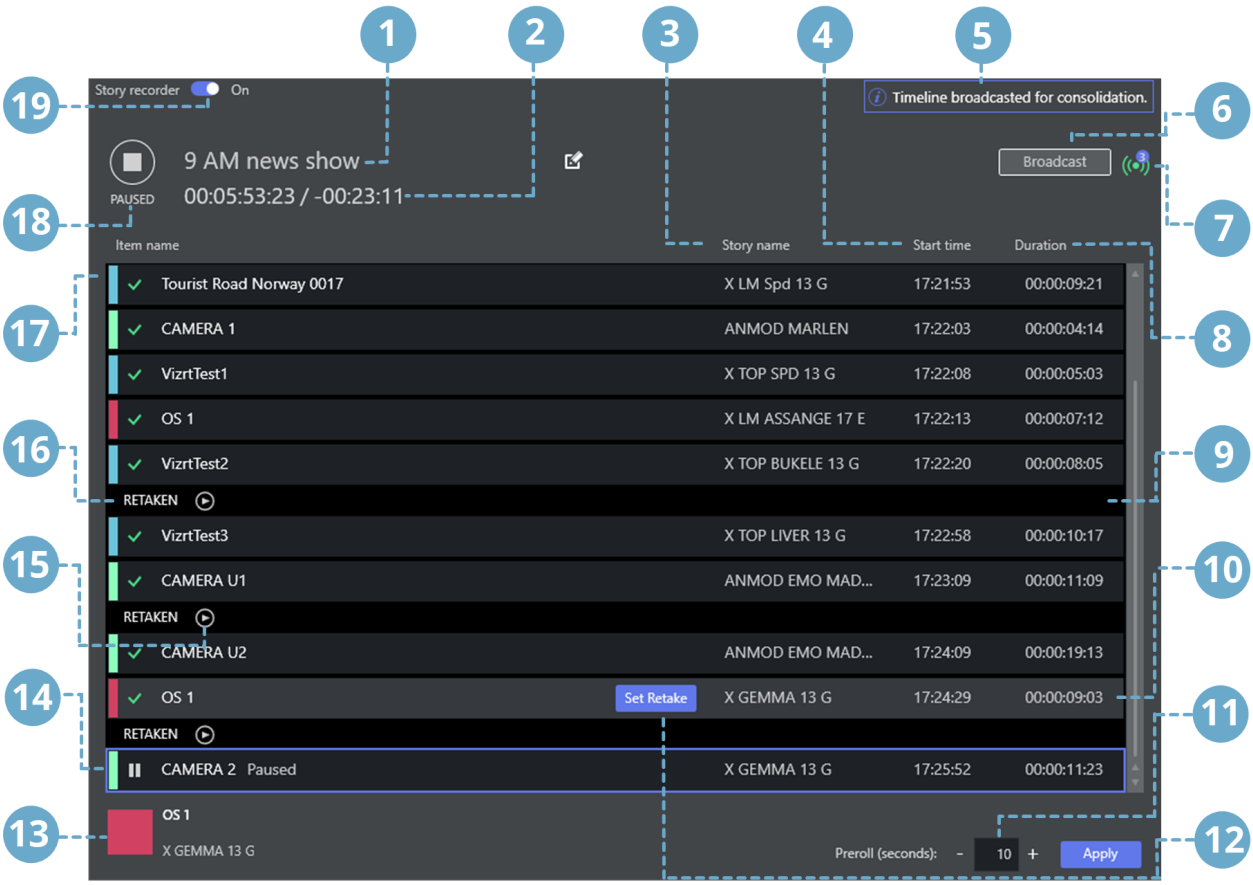
Once Story Recorder mode is enabled, all media fed to PGM is continuously recorded to the video server.

> **Note**: Continuous recording may consume considerable disk space on the video server. See Troubleshooting and Housekeeping for retention settings.

From this point on, any story items taken on air (using the standard Take Next, default **F12**) are recorded and displayed in the Story Recorder panel, item by item.

> **Info**: The video server records continuously from the moment Story Recorder mode is enabled - there is no start and stopping recording at each story or item boundary. Clips are *cut* from this continuous recording and EDLs are *generated* using the frame-accurate timecodes captured by Story Recorder. A rendering or transcoding step is always required to produce the final clips, regardless of workflow.

> **Tip**: If Story Recorder fails to start up, refer to section Troubleshooting and Housekeeping.



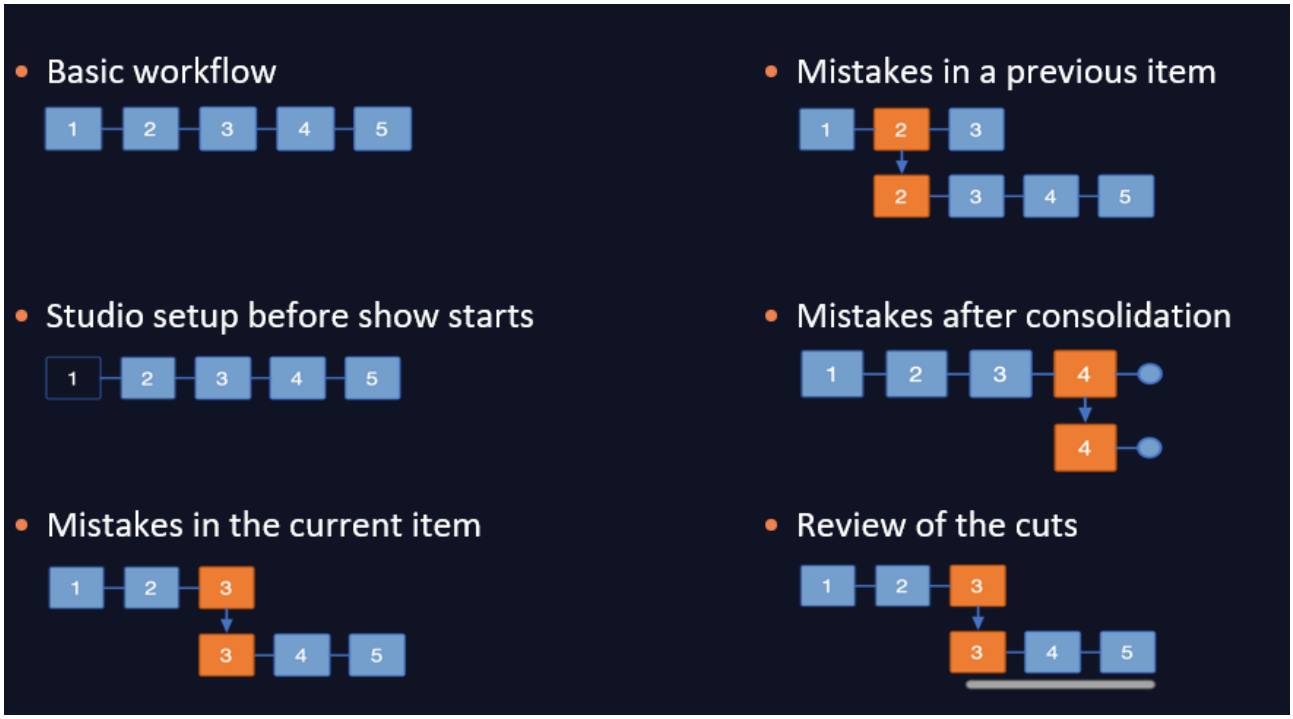
Feature	Explanation
1	**Name of the final clip show and the associated EDL files**: By default, this is the rundown name or provided from NRCS. The user can edit as required.
2	**Recording Time/ Show Countdown** *Recording Time*: When timeline is running, this is continuously growing, equal to the sum of the already recorded sections plus the duration of the current recording. When timeline is paused, this is static, equal to the sum of the recordings prior to the selected retake point. *Show Countdown*: Displays the difference between the planned duration as defined in NRCS and the Recording Time.
3	**Story name**: The story slug, as provided in the NRCS.
4	**Start time**: Timestamp when the item was taken to PGM (wall-time) HH:MM:SS .
5	**Notification**: Operational information and warning/errors.
6	**Broadcast**: Send the stitched show to one or several pre-configured destinations for transcoding (consolidation).
7	**Broadcast count:** Number of successful broadcasts.
8	**Duration**: Story item onair time in HH:MM:SS:FF (frame count only shown after the item is paused/ recorded).
9	**Separator**: Indicates a retake.
10	**Selected row**: The row with focus is colored gray-on-black.
11	**Pre-roll duration**: Pre-roll duration adjustment (seconds). Press **Apply** to set a non-default (10 seconds) value.
12	**Retake point selector**: Arbitrary retake point. This button is displayed when the row is selected and when clicked, it changes the retake point.
13	**Pause indicator**: Displays which story item is currently paused
14	**Blue border**: Retake point indicator. The item that will be taken after pause has a blue border.

Feat ure	Explanation
15	**Cut review player**: Play button is displayed when a snippet clip around the cut point becomes available for review.
16	**Retake marker**: Indicates where an item has been retaken.
17	**Item name**: The item name, that corresponds to a Viz Mosart template and variant
18	**Record | Pause**: Story recorder record and pause button and status indicator.
19	Story Recorder **On** | **Off** switch.

Show Clip Production with Pause and Retake

In this *Show* workflow, the entire show is recorded and a `.tsedl` EDL file is broadcast to a Vantage transcoder, which stitches all recorded segments into a single final clip. The operator can pause, make corrections, and resume recording from a selected retake point.

The following diagram illustrates common recording scenarios. Blue items are recorded segments; orange items indicate retake points.



The sections below describe three common use cases for supporting error-free production and recording of a Viz Mosart-driven show.

- Recording a Show as a Single Clip
  This mode assumes that the entire show was made error-free and Story Recorder has captured the required scheduled performance.
- Pause-Fix-Retake Mode
  An improvement has been identified, and the show is rolled-back to a chosen point, to take again.
- Broadcasting an Alternative Version of the Show
  Sending the recorded stories to a publishing point, either before the show is complete, or with additional stories appended later.

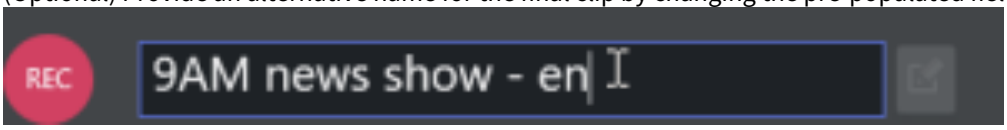
Tip: For a quick guide to what is displayed in Story Recorder, refer back to the illustration The Story Recorder Panel.

Recording a Show as a Single Clip

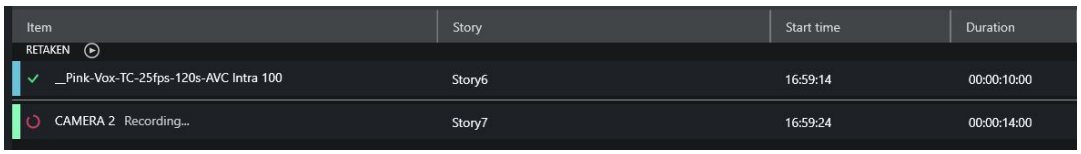
This mode assumes that the entire show was made error-free and Story Recorder has captured the required scheduled production.

To record a show as a single clip

1. Ensure the rundown containing the show's story rows is loaded in Viz Mosart.
2. If the rundown begins with a Break template for initializing equipment, make sure this entire story is executed before enabling Story Recorded mode.
3. From the Story Recorder panel, switch on Story Recorder mode (described above).
   After a few seconds, the **REC** button is red, indicating the system is ready to record your show.
4. If Step **2** above was performed, then execute Take Next to continue with the next story item in the show (this will be the first recorded story item).
   Otherwise begin the show by taking at the first story item of the show.
   For each story item taken, a row is presented in the **Story Recorder** panel, the most recent indicated with *Recording...*
5. When the show is over, an instruction in the last template of the show automatically pauses the recording (see section **Automatic Pause at Show End** of topic *Story Recorder Mode* in the *Viz Mosart Administrator Guide*).
   If an automatic pause has not been set, click the **PAUSE** button in Story Recorder panel or the Pause keyboard shortcut.
6. (Optional) Provide an alternative name for the final clip by changing the pre-populated field in the SR panel:

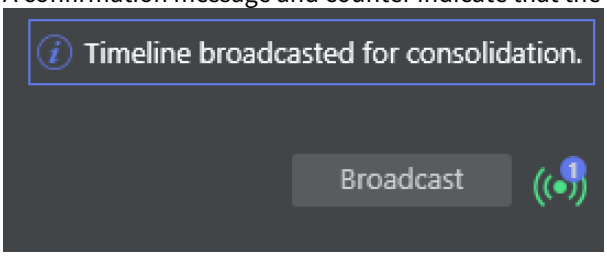


[IMAGE PLACEHOLDER] The editable final clip name field at the top of the Story Recorder panel.



Info: Viz Mosart pre-fills this field from the NRCS when a name is configured there, and otherwise from the rundown name. Anything you type here overrides it. See Show Clip Production Setup for how the name is configured.

7. Click the **Broadcast** button.
   A confirmation message and counter indicate that the clip has been sent for stitching.



Info: The counter indicates the number of EDLs currently being stitched on the EDL transcoder.

- Each time you click **Broadcast**, a new task is sent. See also Broadcasting an Alternative Version of the Show below.
- If you accidentally order the same stitching operation multiple times (by pressing **Broadcast** more than once for your required recording), you can cancel the transcoding job by reviewing the job queue on the transcoder. Your System Administrator can assist with this task.
- The counter is cumulative per recording session, and does *not* decrease once the stitching job has finished or been cancelled on the EDL transcoder side.

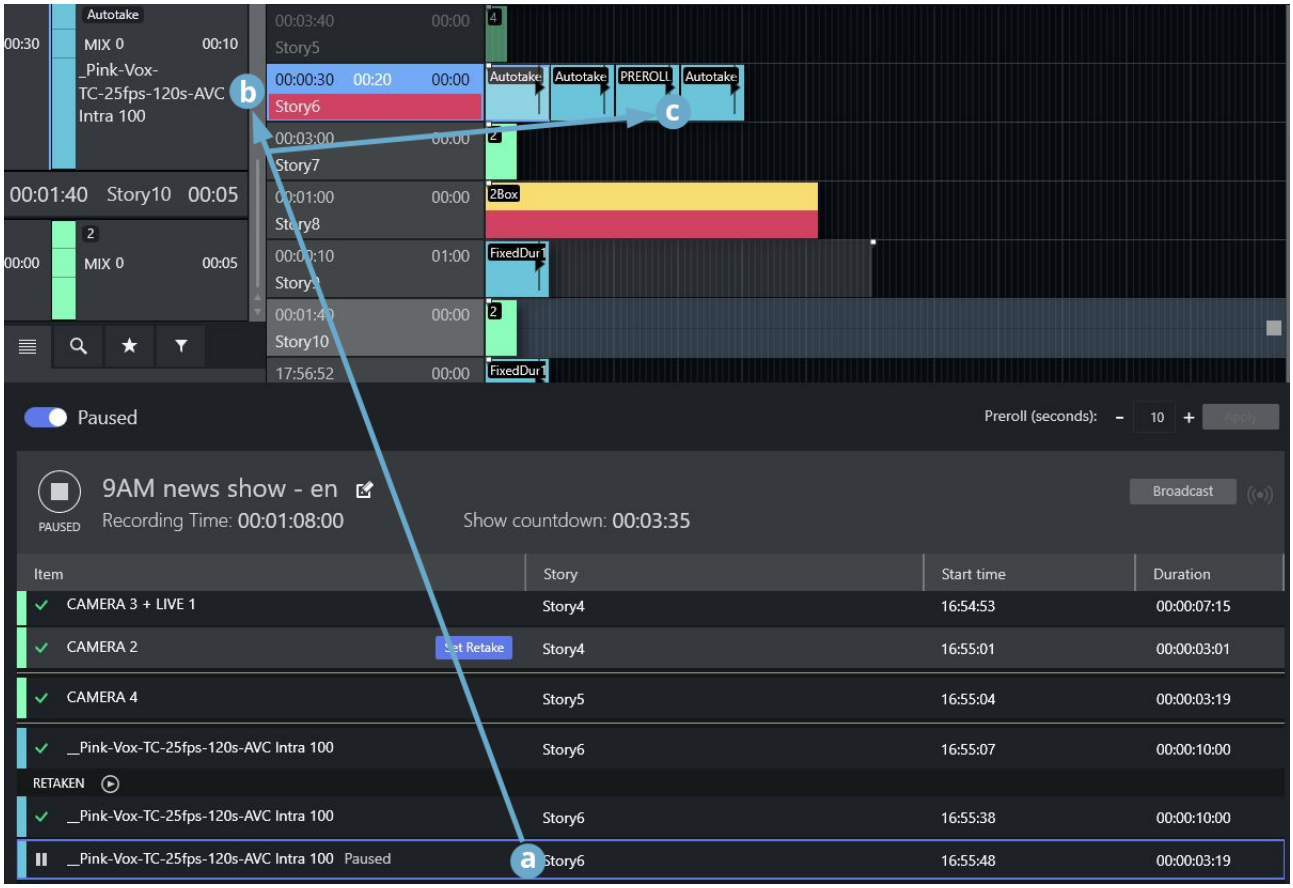
8. Disable Story Recorder with the **On/Off** switch at the top left of the panel.

Pause-Fix-Retake Mode

The standard use of Story Recorder is where you wish to pause an ongoing show recording, make a correction to the production, then resume from the pause.

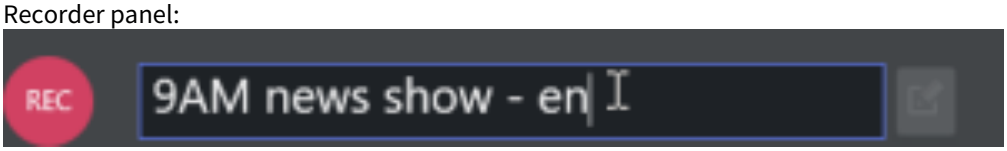
To record a show with pause-fix-retake

1. Initiate recording as described in the first 4 steps of To record a show as a single clip above.
2. Press **PAUSE** to interrupt recording as required. Refer to the screenshot below:
   a. In the Story Recorder panel, the current story item is marked as *Paused* (**a**).
   b. In Rundown view above the Story Recorder panel, the corresponding rundown story title is also blue (**b**).
   c. In the Rundown view, on the story row, a PREROLL template (**c**) is automatically inserted before the paused story item.
3. Make required changes and prepare to recommence recording, by clicking the Story Recorder panel icon, **PAUSED** srP.png .
   A countdown timer cues for retake and recording resumes from the break point.
   Continue as above till the last story line, where an information text warns that the show recording will soon be complete.
   [IMAGE PLACEHOLDER] The PAUSED icon in the Story Recorder panel, with its countdown.



Reviewing the Cut

4. (Optional) After resuming from a pause, a snippet of the previous cut becomes available for review. The snippet appears after the configured **Snippet generation delay**, 15 seconds by default, and runs for the configured **Snippet length**, 5 seconds by default. See Show Clip Production Setup to change either setting. Click the **RETAKEN Play** icon to review the snippet.
5. When the show is over, an instruction in the last template of the show automatically pauses the recording if so configured
   (see section *Automatic Pause at end of Show* under the *Operational Examples > Story Recorder Setup* of the *Mosart Administrator Guide*).
   Otherwise, click the **PAUSE** icon in the Story Recorder panel or the Pause keyboard shortcut.
6. (Optional) provide an alternative name for the final clip by changing the pre-populated field in the Story Recorder panel:

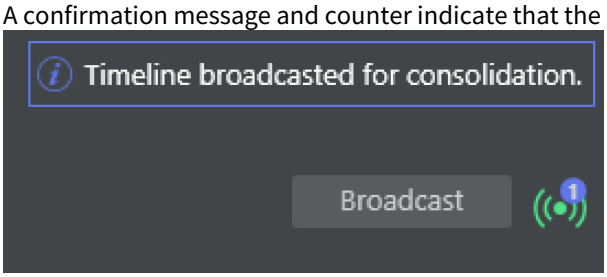


[IMAGE PLACEHOLDER] The editable final clip name field at the top of the Story Recorder panel.

Item	Story	Start time	Duration
0017 Tourist Road Norway	Story3	16:54:09	00:00:12:11
CAMERA 1	Story3	16:54:21	00:00:02:00
RETAKEN			
CAMERA 3 + LIVE 1	Story4	16:54:53	00:00:07:15
CAMERA 2	Story4	16:55:01	00:00:03:01

Info: Viz Mosart pre-fills this field from the NRCS when a name is configured there, and otherwise from the rundown name. Anything you type here overrides it. See Show Clip Production Setup for how the name is configured.

7. Click the **Broadcast** button.
   A confirmation message and counter indicate that the clip has been sent for stitching (consolidation).



Info: The counter indicates the number of EDLs currently being stitched on the EDL transcoder.

- Each time you click **Broadcast**, a new task is sent. See also Broadcasting an Alternative Version of the Show below.
- If you accidentally order the same stitching operation multiple times (by pressing **Broadcast** more than once for your required recording), you can cancel the transcoding job by reviewing the job queue on the transcoder. Your System Administrator can assist with this task.
- The counter is cumulative per recording session, and does *not* decrease once the stitching job has finished or been cancelled on the EDL transcoder side.

Rolling Back in Time at Retake

The smallest rollback is a Viz Mosart primary *story item*. A story row usually comprises several story items, each item defined with a Viz Mosart primary template.
So for example, if your running show is executing item *three* of the four primary items that make up the story, then the pause will cause a rollback to the start of *item three*.

A retake point is always at the ***start* of a primary item**, regardless of where the pause was made.

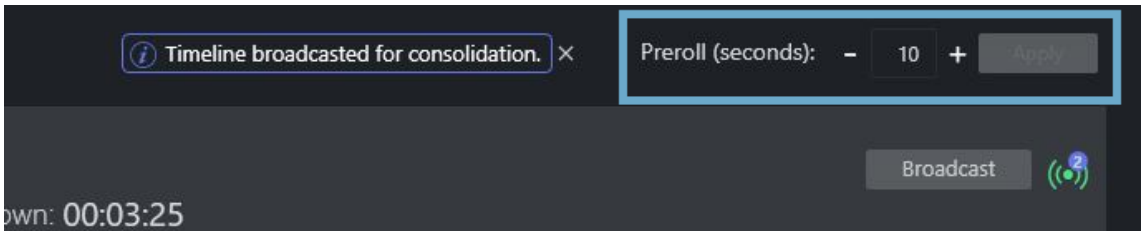
Note: Currently, it is not possible to *replace* a specific recorded story *item*. All the recorded story items after the retake point, including the story item set as retake point, must be re-done once the show is restarted after a pause.

Pre-roll

When you pause a story, Viz Mosart automatically inserts a pre-roll event in the story row, before your paused story item.

The pre-roll gives the equipment time to return to a known state (camera positions, graphics, media clips, fader levels), and it cues the talent to begin again. Your administrator sets the minimum duration, usually 10 to 15 seconds, long enough for the video server to be ready to record again. See Show Clip Production Setup for how that minimum is chosen.

- You can configure the pre-roll time from the Story Recorder panel:



Broadcasting an Alternative Version of the Show

Broadcasting means auto-creation of a show-level EDL, from the list of story rows in the Story Recorder panel. This EDL is then fed to the EDL transcoder (Vantage) watch folder. The transcoder picks up the EDL file and initiates the job of stitching the clips (that represent the story rows), into a single, show clip. This single clip is used for playout.

- You can broadcast at any time, during a paused recording, and then continue recording.
  This can for example be when producing a trailer, a second language variant of the show, or when in the afternoon you want only to change the last couple of stories from the morning's recording.
- Similarly, you can append additional rundown (story) rows to a completed, paused show.
  Use the **Tools > Arrange rundowns** menu, as described in the topic Arrange Rundowns.

In these scenarios, you must always create a Retake point in the story *before* the change or addition.

Story Clip Production and Publishing

In this workflow, stories marked for recording in the NRCS are individually recorded and produced as clips, either during a live broadcast or in an offline session. Clip production is triggered automatically - by a story boundary or by disabling Story Recorder mode. The recordings are then submitted to the configured output (TriCaster export or Mimir render job).

Info: Even when recording a single story, the clip must still be rendered from the continuous recording. The story clip is not a direct capture, it is cut using the frame-accurate in and out timecodes, subsequently rendered by the configured output.

The sections below describe three common use cases for story clip production:

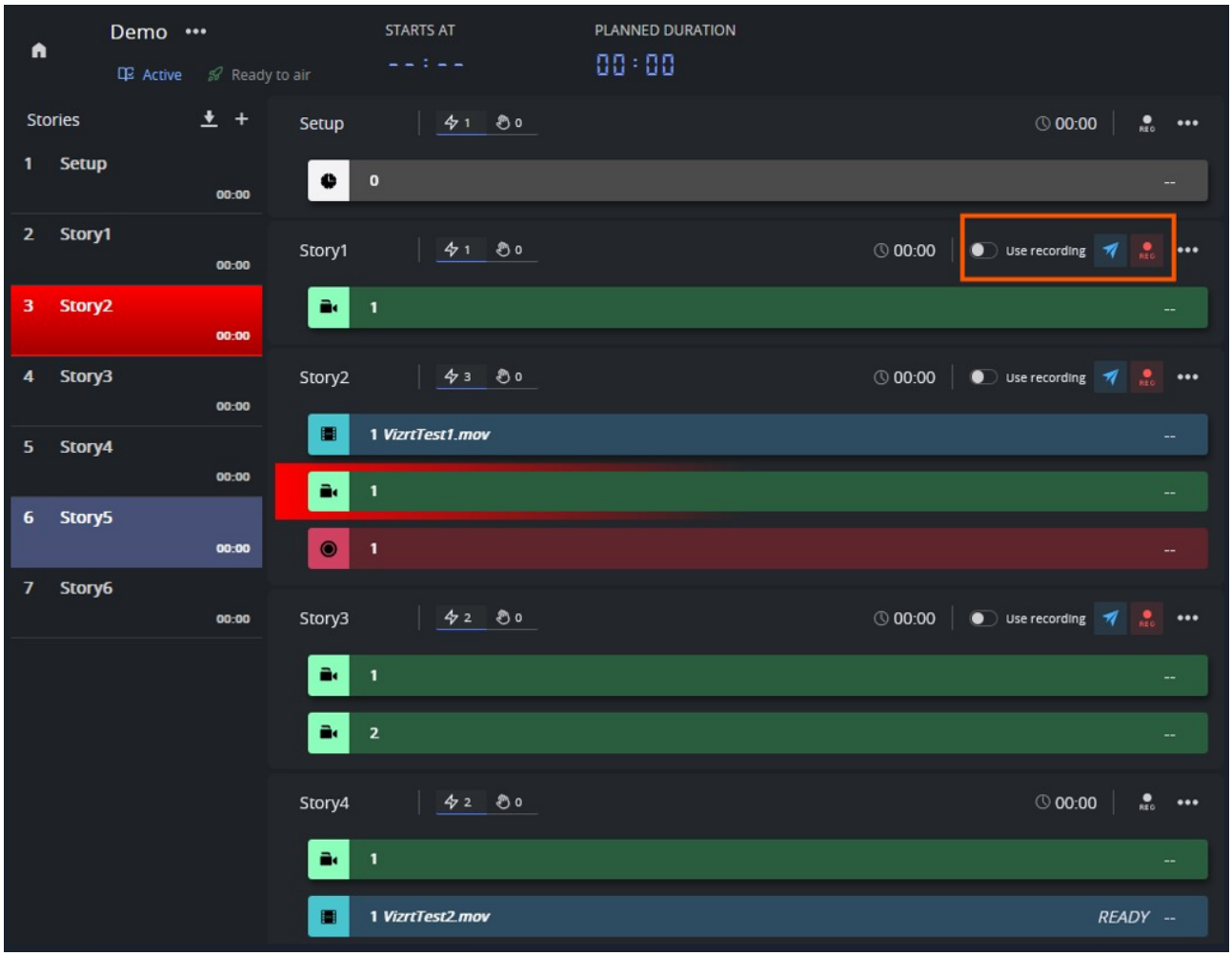
- **Producing clips during a live broadcast**
  Clips are produced automatically at each story boundary while the show runs live.
- **Recording a single story**
  A single story is recorded and clip production is triggered by disabling Story Recorder mode.
- **Offline recording with stories back-to-back**
  Multiple stories are recorded in sequence, with the operator controlling the pace between stories using a template control command.

Stories Marked for Recording

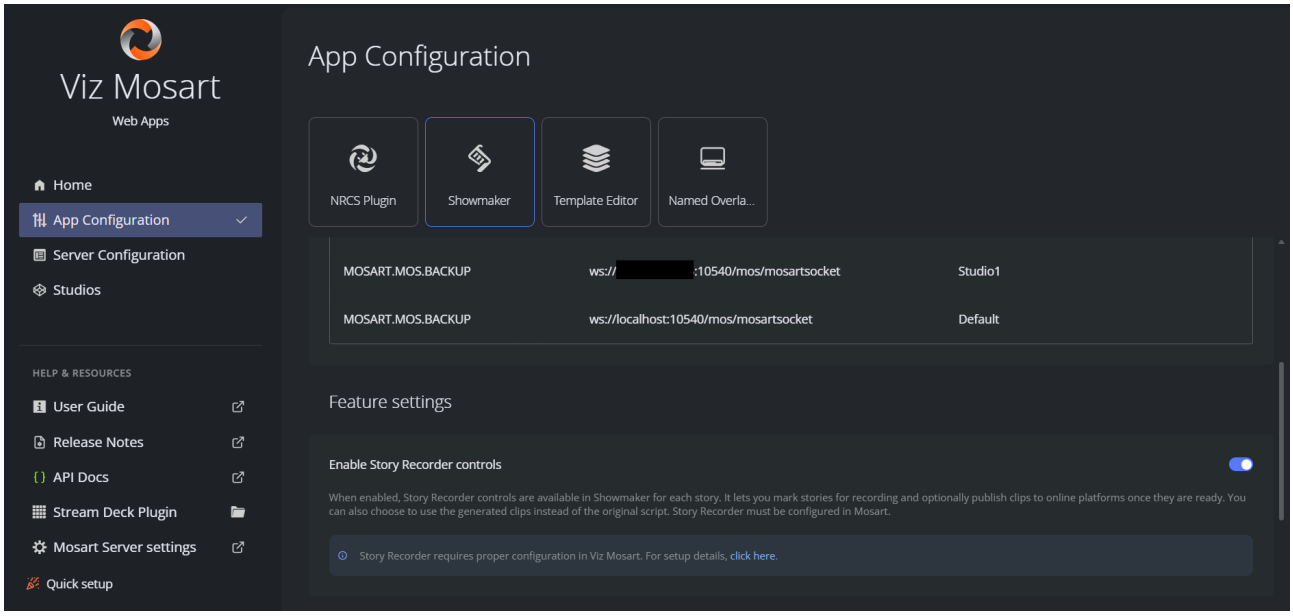
For the *story* clip production workflow, stories must be tagged for recording in the NRCS *before* the show. Any NRCS that can provide the required metadata fields on the MOS story object can be used with this workflow. Saga and Showmaker offer built-in integration; other NRCS systems can be configured via Newsroom Settings field mapping.

See Story Recorder Setup for the NRCS metadata requirements and configuration details.

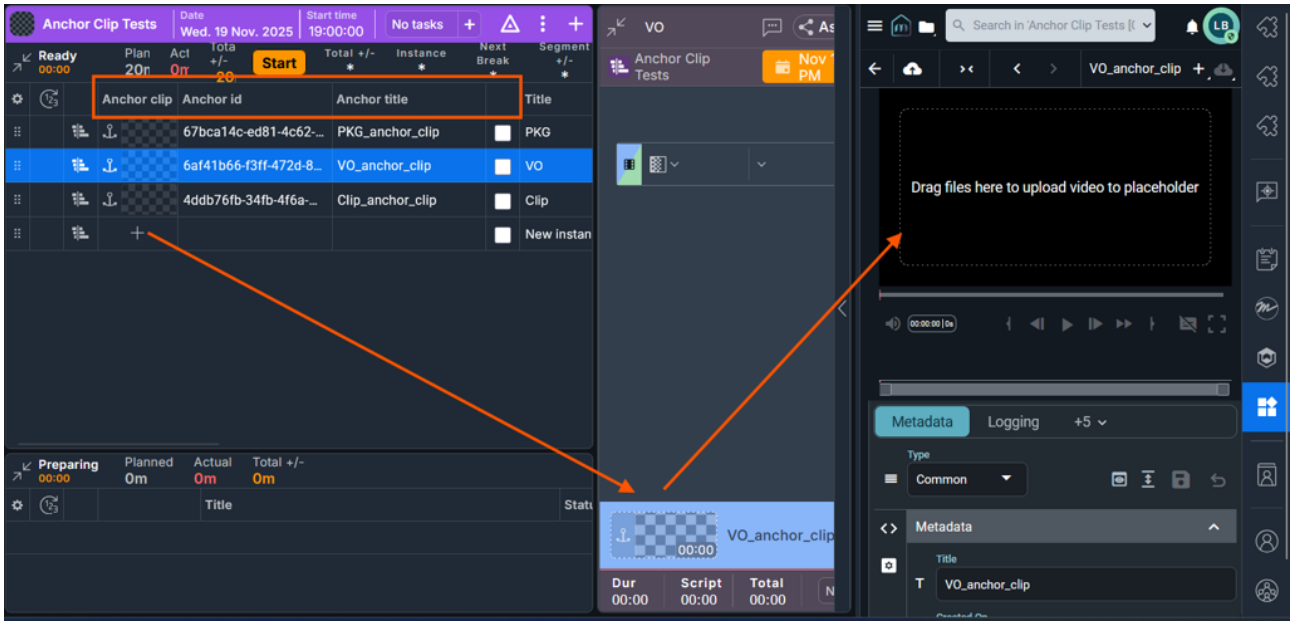
Here are how stories can be marked for recording in Showmaker:



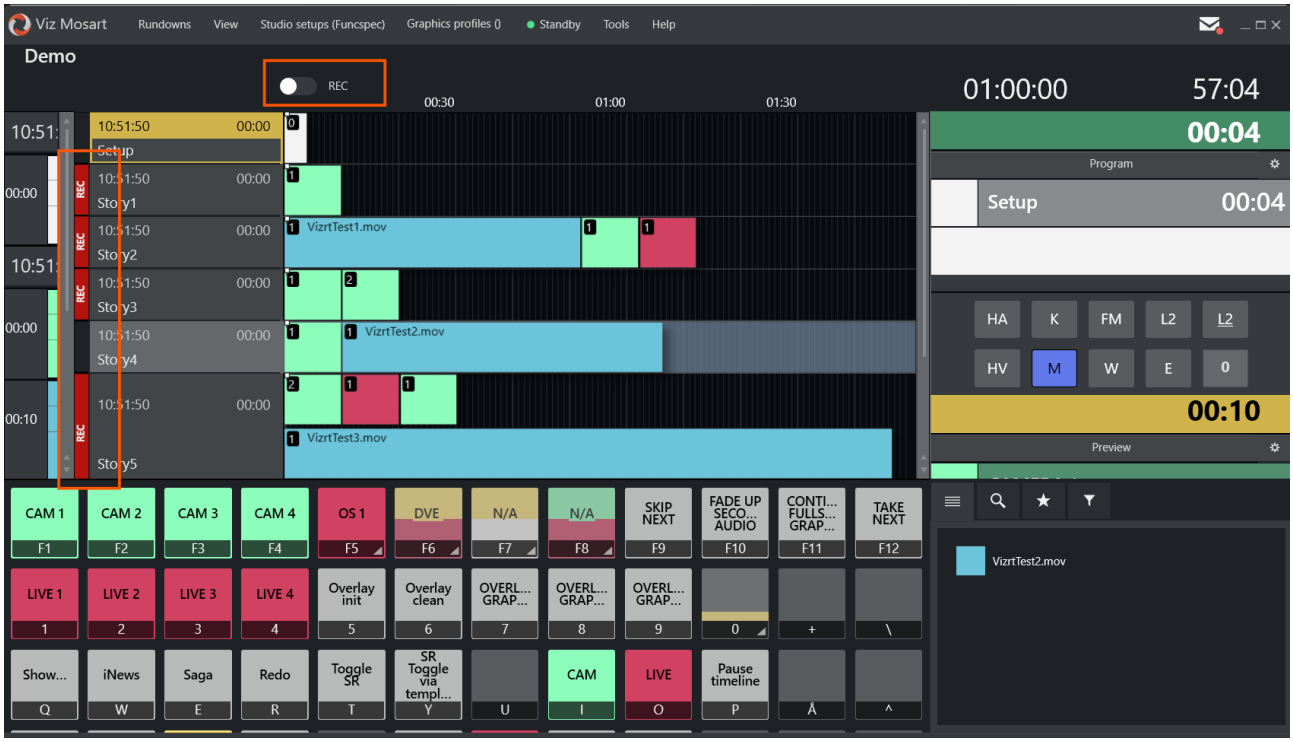
Note: To have these Story Recorder options on a story in Showmaker, you need to enable Story Recorder controls from *Mosart Web Config Tool* > *App Configuration* > *Showmaker*.



And here is a story in Saga with Anchor clip placeholders (left), each linked to a Mimir clip placeholder (right), ready to receive recorded clips from Story Recorder:



When a rundown containing tagged stories is loaded in Viz Mosart, these stories are shown with a **REC** indicator in the main timeline view. The Story Recorder mode toggle at the top of the timeline also indicates whether there are stories to be recorded.



The **REC** indicators means that the tagged story is configured for clip production. When Story Recorder mode is enabled and a clip output (renderer/transcoder) is configured, a clip will be produced for this story. The indicator does *not* mean the story is currently being recorded, and no clip production occurs unless Story Recorder mode is enabled.

Clip Production Workflows

Producing clips during a live broadcast

In this use case, story clips are produced automatically as the show runs live. No manual intervention is needed, clip production is triggered at each story boundary.

1. In the NRCS, ensure the relevant stories are marked for recording.
2. Load the rundown in Viz Mosart. Verify that tagged stories show **REC** indicators in the timeline.
3. Enable Story Recorder mode (using any of the methods described in Managing a Story Recorder Session).
4. Run the show: Take stories as usual, with Take Next (default *F12*).
    - When the first item of a new story is taken on air, the previous story's recordings are automatically submitted for clip production.
    - Stories that are *not* marked for recording are silently skipped.
5. When done, disable Story Recorder mode. Any remaining recorded story is submitted for clip production automatically.

Recording a single story

In this use case, you record only one story and then stop.

1. In the NRCS, ensure the relevant story is marked for recording.
2. Load the rundown in Viz Mosart and navigate to the story. Verify the story displays a **REC** indicator in the timeline.
3. Enable Story Recorder mode.
4. Run the story, taking items as usual.
5. At the end of the story, disable Story Recorder mode. This triggers clip production for the recorded story.

> **Info**: The story's out-point is determined by the genlock timecode at the moment the Disable command is processed. This may be a few frames after the operator's action (key press or template take), depending on system load.

Offline recording with stories back-to-back

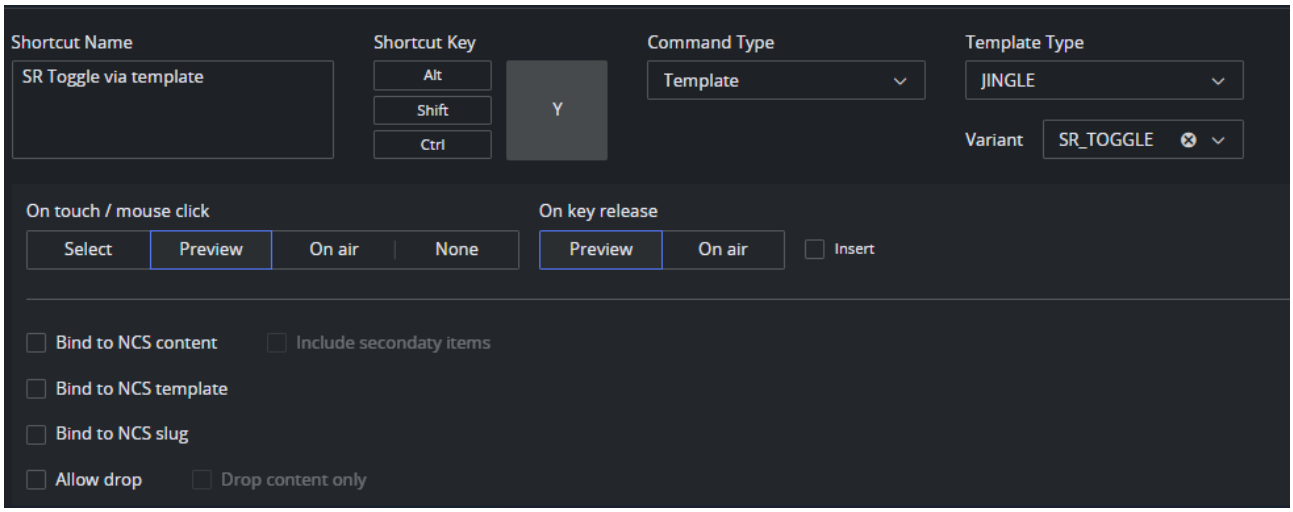
In this use case, you record multiple stories in an offline session where the operator controls the pace between stories For example, allowing the anchor time to prepare or reposition between takes.

This is achieved by using a dedicated template with a `STORY_RECORDER` control command that toggles SR mode *on* and *off* between stories. The template is configured with:

- **Command On Take**: `STORY_RECORDER` with value `TOGGLE` (or `DISABLE`) . This disables SR mode when taken on air, triggering clip production for the completed story.
- **Manual Continue Point**: `STORY_RECORDER` with value `TOGGLE` (or `ENABLE`). This re-enables SR mode when the operator takes the next manual continue point.

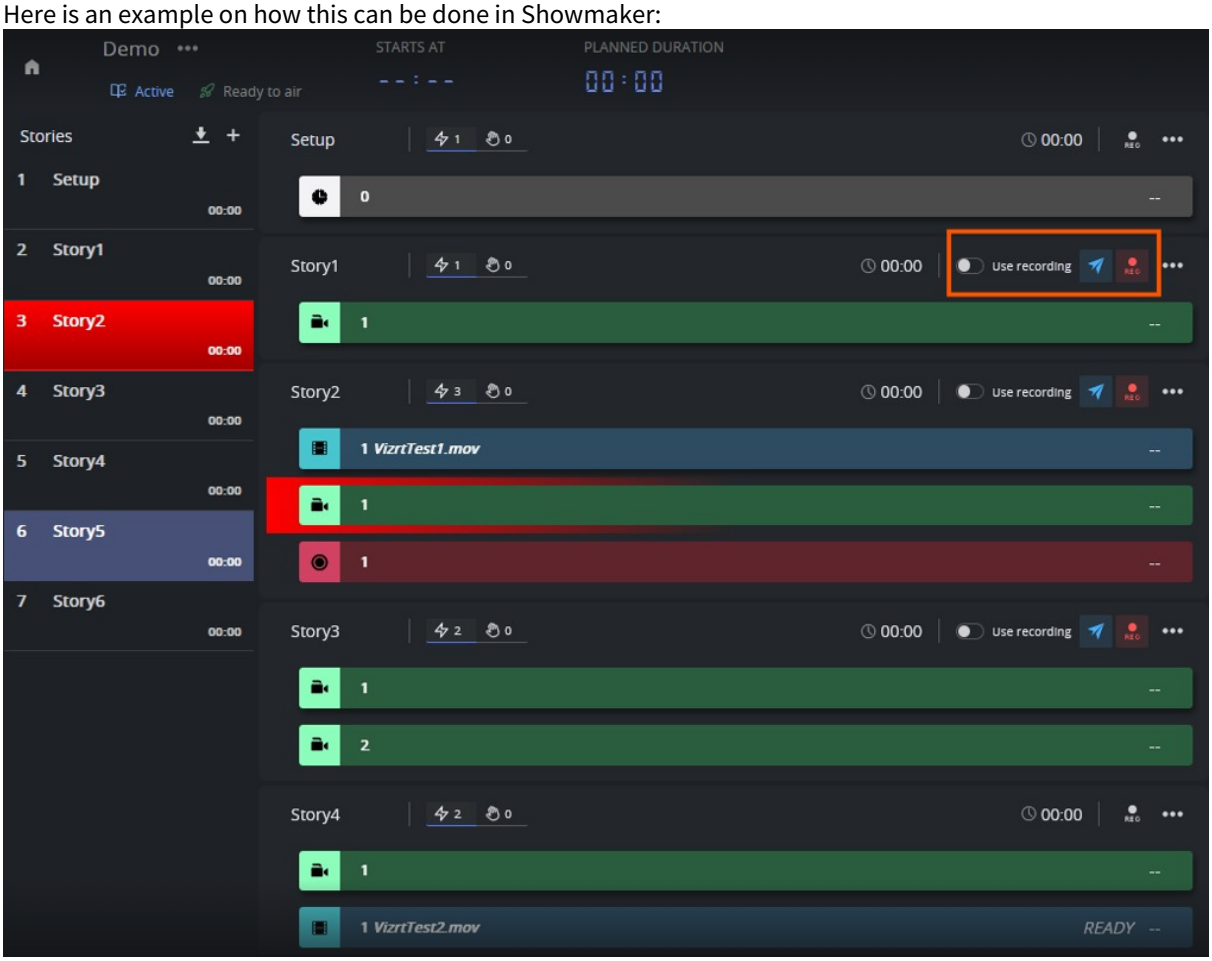
The SR Toggle Template can be executed in two ways:

- **Keyboard shortcut**: Configure a keyboard shortcut to take the template in Preview. The operator recalls this shortcut on the last item of each story, then takes the template on air when ready to end the story.
- **From the NRCS**: Add the SR toggle template at the end of each story in the rundown. The template is then automatically queued as the last item of the story.

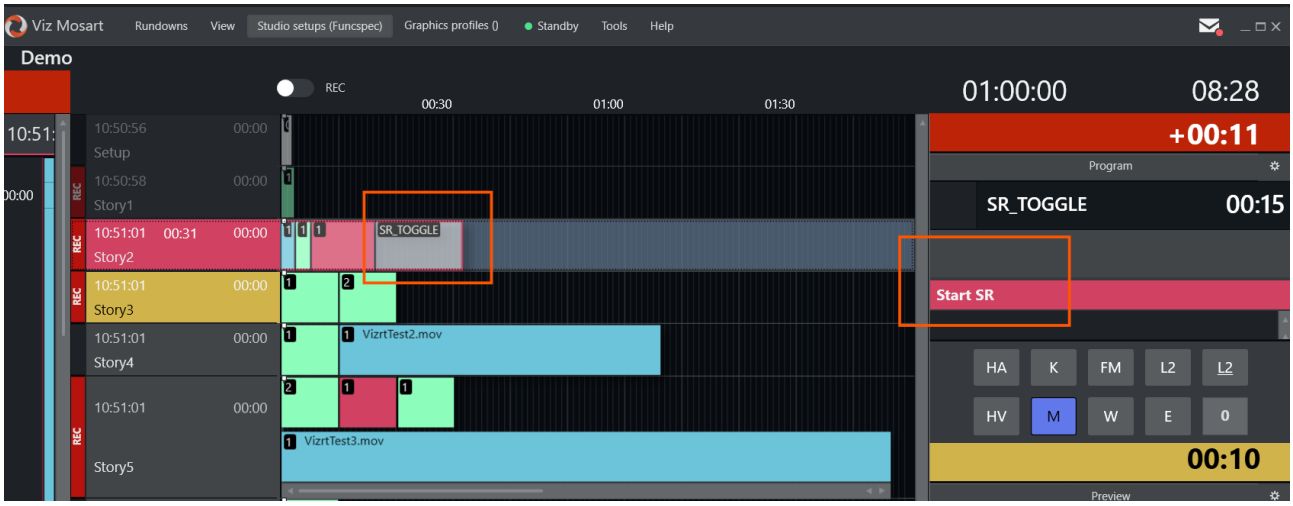


Workflow:

1. In the NRCS, tag the stories for recording.
   Here is an example on how this can be done in Showmaker:



2. Load the rundown in Viz Mosart.
3. Enable Story Recorder mode and take the first story.
4. On the last item of a story, bring the SR Toggle Template into Preview (using its keyboard shortcut, or automatically if placed in the rundown).
5. When the story is finished, take the template on air. This disables SR mode and triggers clip production for the completed story.
   The manual continue point is indicated in the Mosart UI.
6. When ready to continue with the next story (for example, once the anchor has repositioned), press Take Next. This takes the continue point, re-enables SR mode, and recording resumes with the next story.
7. Repeat steps 4-6 for each story.
8. After the last story, disable Story Recorder mode to flush the final clip.



> **Info**: The out-point of the story is determined by the genlock timecode at the moment the Disable command is processed. This may be a few frames after the operator's action (key press or template take), depending on system load.

TriCaster Clip Export

When Story Recorder is configured with a TriCaster output, completed stories are exported as clips using configured TriCaster export presets. Stories not marked for recording in the NRCS are skipped.

- The clip name is based on the clip ID assigned to the story in the NRCS, with a timestamp appended for uniqueness. If the NRCS did not assign a clip ID, a name is auto-generated from the story name and recorder channel.
- If the story has also been tagged in the NRCS for online publishing, the clip is published to the configured online platform (for example, *YouTube*) after export.

See the Viz Mosart Administrator Guide (section *Operational Examples > Device-Specific Operations*), for TriCaster export and publish preset configuration.

Mimir Clip Production

When Story Recorder is configured with a Mimir output, completed stories are submitted as render jobs to Mimir. Stories not marked for recording in the NRCS are skipped.

- When Story Recorder mode is enabled, Mosart creates recording clip placeholders in Mimir for each configured recorder.
- At story completion, a render job is submitted to Mimir using the configured preset and Kelda configuration.
- The clip ID assigned to the story in the NRCS is used as the target item (Anchor clip) in Mimir.
    - On the first recording, the clip is rendered into the existing anchor item placeholder.
    - For repeated recordings of the same story, a new version is created under the same anchor item, preserving the version history.
      Mimir then updates the NRCS (e.g. Saga) with the new anchor clip ID, so subsequent recordings reference the latest version.

See the *Story Recorder Setup* section of the Viz Mosart Administrator Guide for Mimir device, preset, and Kelda configuration.

Using Recorded Story Clips

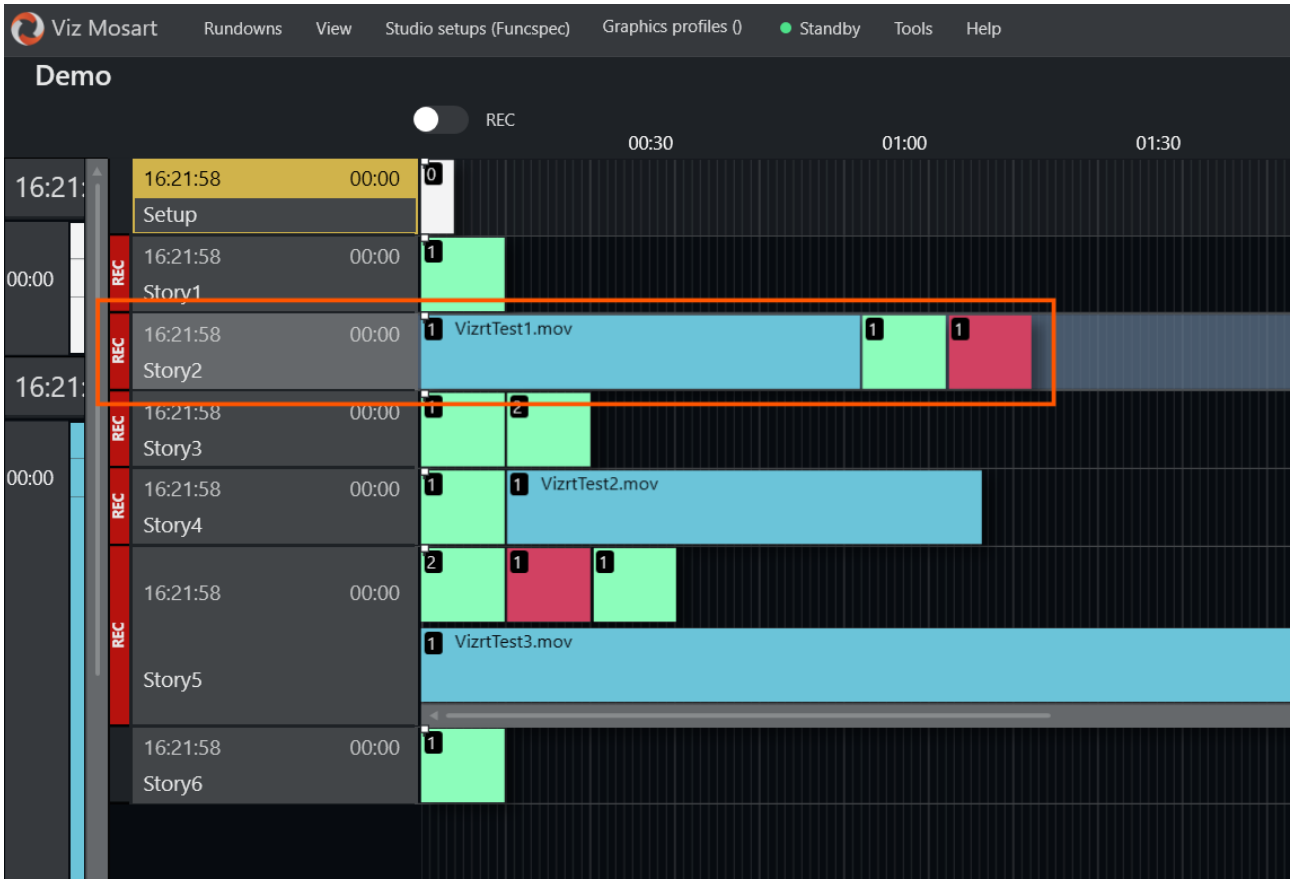
Once a story clip is produced, it can be used in several ways:

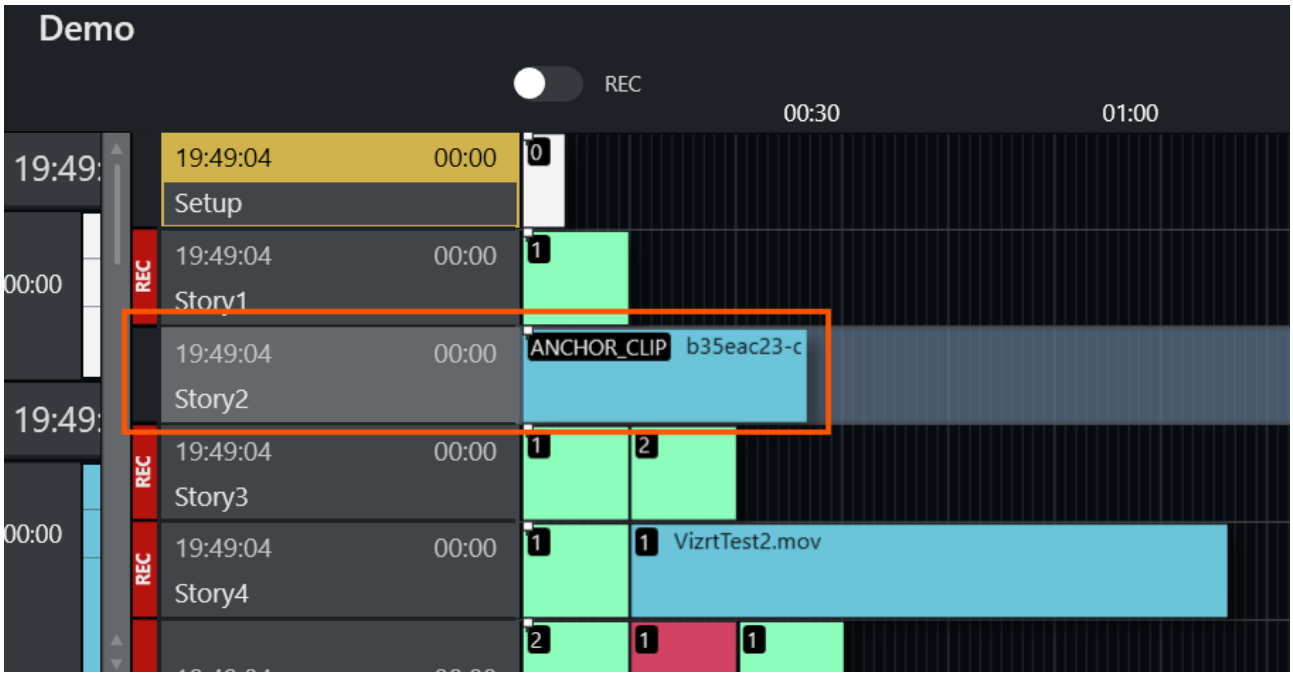
- **Replace the original story**
  The NRCS can be configured to play the recorded clip instead of the original scripted content.
  This is useful when a pre-recorded version of the story should air in place of the live segment.
- **Reuse in other rundowns**
  The recorded clip is available as a regular media asset and can be inserted into any other rundown.
- **Add graphics to a clean feed**
  If the clean feed (without burned-in graphics) was recorded, new graphics can be overlaid on the clip, for example for rebranding or a different-language broadcast.

Switching between original and recorded clip in the NRCS

As an example of an NRCS, with *Saga*, the producer can choose per story whether to use the original script or the recorded clip version. When the recorded clip is selected for a story, Saga sends the clip to Viz Mosart using a template of type `PACKAGE` with variant `ANCHOR_CLIP`.

> **Info**: This template must first be created in Viz Mosart's Template Editor before it can be used.

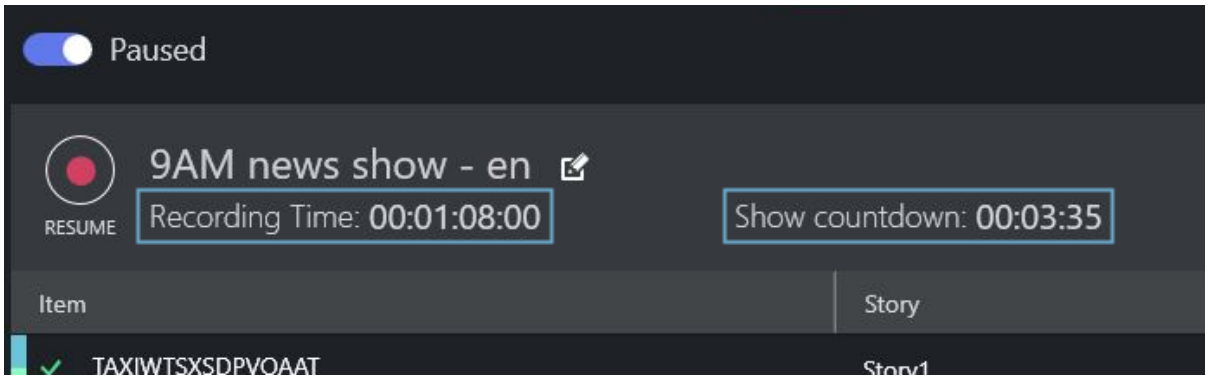

Viz Mosart
Rundowns
View
Studio setups (Funcspec)
Graphics profiles ()
Standby
Tools
Help
Demo
REC
00:30
01:00
01:30
16:21:58 00:00
Setup
Story1
Story2
Story3
Story4
Story5
Story6
VizrtTest1.mov
VizrtTest2.mov
VizrtTest3.mov


Demo
REC
00:30
01:00
19:49:04 00:00
Setup
Story1
Story2
Story3
Story4
ANCHOR_CLIP
b35eac23-c
VizrtTest2.mov

Timing Details

The Mosart UI has been extended with support for Story Recorder-related timing fields.

The Story Recorder panel displays the following new timing details:



The **Recording Time** field starts at zero when a Story Recorder session is started, and

- When the timeline is *running*, this counter is continuously growing, equal to the *sum of the already recorded sections + the duration of the current recording*.
- When the timeline is *paused*, this counter is static, equal to the *sum of the recordings prior to the selected Retake point*.

The **Show Countdown** field displays the *difference* between the *planned duration* as defined in the NRCS and the *Recording Time*.

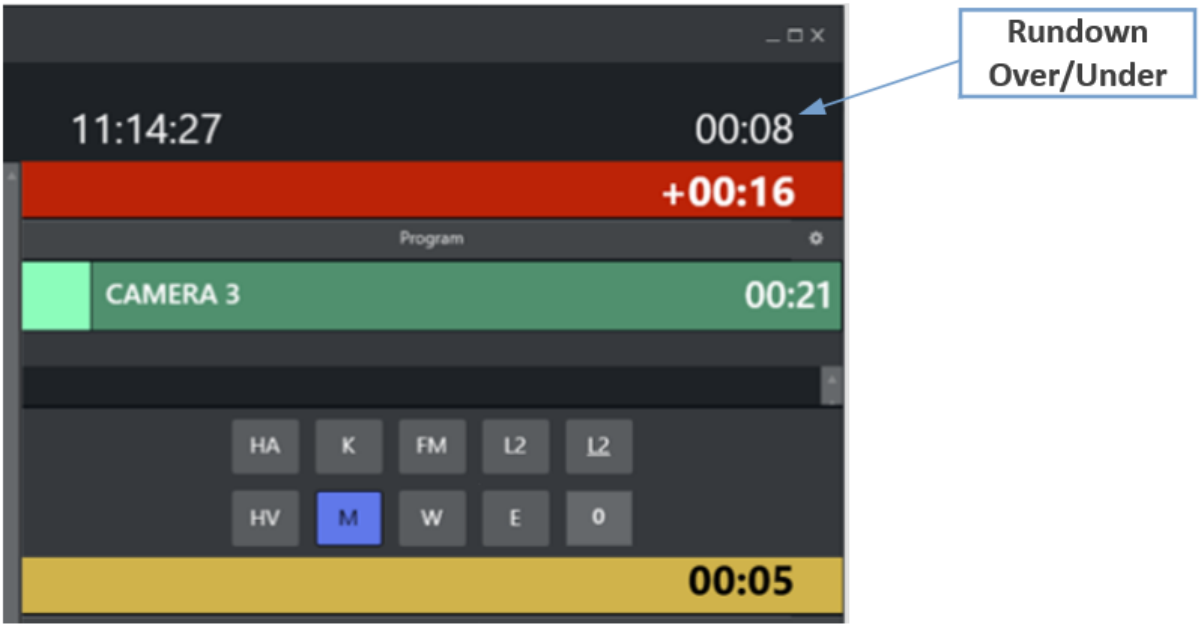
- When the timeline is running, this counts down.
- When the timeline is paused, this is static.

If *Recording Time* exceeds *Show Planned Duration*, then *Show Countdown* becomes negative and will be counting up.

The **Rundown over/under** field is an existing field in the Viz Mosart UI, and when Story Recorder mode is enabled, it provides an indication of the executed recordings relative to the remaining rundown and the planned duration of the rundown:

- **Under**: indicates the number of seconds the show must be *extended* to meet the planned duration.
- **Over**: indicates the number of seconds the show exceeds the planned duration.

Copyright © 2026 Vizrt



All these Story Recorder related time fields are also shown in the **Mosart Timing Display** and forwarded to the *Plura Timing System* (using the *Time Request Protocol*).

12.4.6 Story Recorder Troubleshooting

- How Story Recorder Reports Problems
  - Session states
- Message Reference
  - Information
  - Errors
  - Warnings
  - Confirmation dialogs
- Story Recorder Will Not Enable
  - Story Recorder did not start: the Mosart Logging Service is not responding
  - Recording ports not found
- Recording Does Not Start or Items Are Missing
  - Recording not starting
  - Story items do not appear in the Story Recorder panel
- Timecode and Genlock Problems
  - Genlock mode is not enabled
  - Unreliable genlock
  - Time sync issue between the video server and Viz Mosart
  - All clips are offset by the same amount
  - Only the first story is offset
  - The frame rate does not match the video server
  - Viz Mosart is reading the wrong recorder output
  - TriCaster start of media is UTC
  - TriCaster stops reporting start of media
- Clip Production and Publishing Failures
  - Mimir: timeout waiting for ingest
  - Mimir: other failures
  - TriCaster export
  - Vantage and EDL output
  - Snippet review clips
- Pause and Retake Errors
  - Pause is ignored
  - Invalid retake position
  - Retake point replaced by an earlier retake
  - Error when pausing the story
  - Retake requested too early
- Naming of the Final Clip
- Housekeeping and Disk Space
- Limitations
- Collecting Data for Vizrt Support
  - Files
  - Settings to report
  - Questions to answer

Copyright © 2026 Vizrt

This section covers problems reported by Story Recorder, what each message means, and how to resolve it. It also lists the data to collect before contacting Vizrt Support.

> **Start here.** Many symptoms that look like Story Recorder faults are timing faults. If clips are produced but their content is offset, or edit points do not land where expected, the cause is usually genlock or timecode rather than Story Recorder. Go to *Timecode and Genlock Problems* below, and then to the Frame Accurate System Operations section.

How Story Recorder Reports Problems

Story Recorder shows messages in two places, and both show the same text:

- The Story Recorder panel, at the top of the panel.
- The toolbar strip in the Viz Mosart timeline view.

The toolbar also shows the live Story Recorder status next to the on/off switch: *SR Off*, *SR Starting...*, *SR Prerolling...*, *SR Ready!*, *SR Paused*, and *SR Stopping...*. The toolbar and the Story Recorder tool window always show the same status.

Messages carry a severity of Information, Warning, or Error. Confirmation dialogs appear separately, as modal windows.

> A long message is shortened in a narrow window. Hover over it to read the full text, and the complete text is always written to the Viz Mosart logs.

Session states

Story Recorder moves through the following states. The state determines which operations are accepted, so it explains several of the messages below.

State	Meaning
Disabled	Story Recorder is off. This is also the state forced at startup, so a stale state is never carried over.
Booting	Story Recorder is starting up and checking prerequisites. Genlock warnings are raised here.
Prerolling	Recorders are being prepared and started.
Recording	Recording is running. The REC indicator blinks bright red.
Pausing	A pause has been requested and the recorders are being stopped.
Paused	Recording is paused. A retake point can be selected and the recording can be delivered.

Copyright © 2026 Vizrt

State	Meaning
RundownInRehearsal	A rundown was reloaded into rehearsal while recording. Recording is paused and no clips are produced.
ShuttingDown	The session is closing and any final publishing is running.

Message Reference

Values shown as *<port>* or *<folder>* are filled in at runtime.

Information

Message	Meaning
Story Recorder initialized.	The Story Recorder log adapter started successfully. No action needed.
Using recorders *<name (server:port)>*, ... for Story Recorder	Lists the recorder ports that the session will use. Check that the expected ports are listed.
Delivered to *<outputs>*.	The result of pressing **Deliver**: lists each output and what it received, for example *Delivered to TriCaster (4 stories), Mosart EDL (6 items, 4 stories, 1 show) and Vantage.* The counts cover the whole session, not only the last press.

Errors

Message	What it means and what to do
Story Recorder did not start: the Mosart Logging Service is not responding. Check that the service is running on the Mosart server, then switch Story Recorder on again.	The service that hosts Story Recorder did not answer. See *Story Recorder Will Not Enable* below.
Story Recorder did not start, although the Mosart Logging Service is responding. Switch Story Recorder on again, and check the Mosart logs if it keeps failing.	The service is running, but the Story Recorder adapter did not complete its start. Switch Story Recorder on again. If it keeps failing, check the Mosart logs and that the Logging Service points at the Server installation.

Copyright © 2026 Vizrt

Message	What it means and what to do
Failed to handle Story Log Adapter event.	An event from the log adapter could not be processed by the client. Check the logs and report to Vizrt Support.
Recorder *<port>* did not respond to the prepare/start command. Check the video server.	The video server did not answer within the timeout. Check that the server is reachable and the port is not in use elsewhere.
Failed to prepare recording for port: *<port>*.	The video server rejected the prepare command. Check the port configuration and free disk space on the server.
Failed to start recording for port: *<port>*.	The video server rejected the start command. Check the port state and the recording directory.
Failed to create recording placeholder item in Mimir.	The Mimir create-item call failed. Check the Generic REST device configuration and Mimir availability.
Failed to validate the following properties: *<list>*. Restart Mosart Server Log Service after the fix.	One or more Story Recorder settings are invalid. Correct the listed settings, then restart the Mosart Server Log Service.
No start timecode received from the video server (port *<port>*).	The start of media request returned nothing. Edit points cannot be calculated reliably. See *Timecode and Genlock Problems*.
The video server returned no valid start timecode (port *<port>*).	The start of media value was present but not usable. See *Timecode and Genlock Problems*.
No recorders configured for Story Recorder	No record port is available. Add a record port to the current salvo in AV Automation. See *Recording ports not found* below.
Waiting for recorder to pause. Please try again.	A retake was requested while the recorder was still stopping. Wait for the Paused state, then retry.
The recording has not yet started. Please try again after the system has paused.	A retake was requested before recording began. Wait for the Paused state, then retry.
Error while cleaning the snippets folder. *<errorType>*: *<folder>*.	The snippets folder could not be cleaned. Check that the folder exists and that the Logging Service account has write and delete rights.

Copyright © 2026 Vizrt

Message	What it means and what to do
The snippet transcoding process failed.	Viz Coder could not transcode the review snippet. Check that Coder is installed and that the path in `CoderTools.dll.Config` is correct.

Warnings

Message	What it means and what to do
Genlock mode is not enabled.	Story Recorder starts without genlock and uses time of day. Acceptable for testing only. See *Genlock mode is not enabled* below.
Salvo changed while Story Recorder is active. The record port does not change. Reload video ports (CTRL+SHIFT+V) or restart the session to apply it.	A running recording keeps the port it started on. The next recording is prepared on the record port of the current salvo.
Pause ignored: retakes are switched off in the EDL output settings.	**Enable retakes** is off in the EDL output settings, so the pause request is refused and the retake controls are greyed. Switch it on if the workflow uses pause and retake; see Show Clip Production Setup.
The retake point is no longer valid: the take it pointed to was replaced by an earlier retake. Select another retake point.	The selected retake point was superseded by an earlier retake. Select another retake point. The recorded items and EDL entries are kept.
Story Recorder has no output configured. Recordings will be made but nothing will be published.	Raised when Story Recorder is enabled with no clip or EDL output configured. Configure at least one output; see the workflow pages under Story Recorder Setup.
Vantage EDL output is enabled but its local or watch folder is not configured. No Vantage EDL will be written.	Fill in both the local folder and the watch folder in the Vantage output settings; see Show Clip Production Setup.
Nothing to deliver: no output is configured for Story Recorder.	**Deliver** was pressed with no output configured. Configure an output, then deliver again while the session is still paused.

Copyright © 2026 Vizrt

Message	What it means and what to do
Delivered with errors: *<n>* EDL files could not be written. See logs for details.	One or more EDL files failed to write. Check that the EDL folders exist and that the Logging Service account can write to them, then check the logs.
Recorder *<port>* did not respond to the stop command.	The recording may still be running on the video server. Check the port state before starting a new session.
Time sync issue: video server time (*<tc>*) differs from Mosart time from external clock (*<tc>*).	The video server clock and the Viz Mosart clock disagree. Recording is not blocked, but edit points may be offset. See *Timecode and Genlock Problems*.
Issues found related to the housekeeping task to delete the recording clips created by Story Recorder.	Automatic purging of intermediate recordings did not run cleanly. See *Housekeeping and Disk Space*.
Recording cleanup folder not found: '*<folder>*'	The configured recordings folder is missing or unreachable. Variants of this message report access denied and validation errors.
Failed to generate the snippet clip around the cut point.	The review snippet was not produced. The retake itself is unaffected.
Next story will stop the recording	The upcoming story ends the recording. Informational, shown ahead of the stop.

Confirmation dialogs

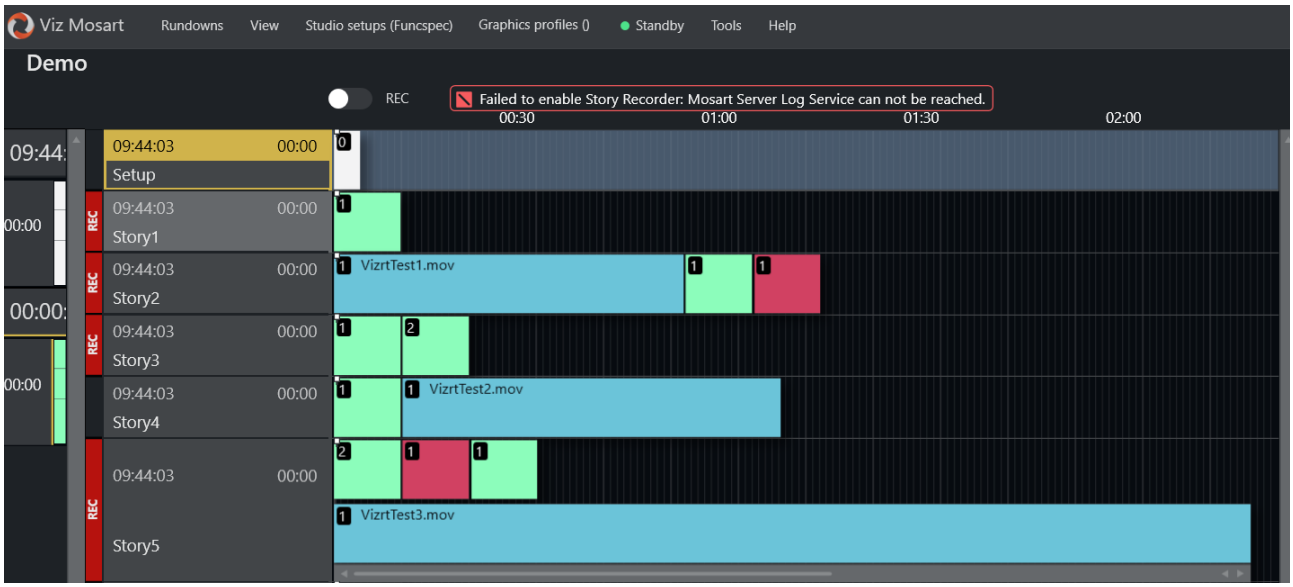
Dialog	When it appears
End the Story Recorder session? Recordings stay on the video server and remaining stories are sent to their outputs. If you have not pressed Deliver in the Story Recorder tool window, the show EDL is not created, and it cannot be created after the session ends.	When disabling Story Recorder, if confirmation on disable is switched on.
Are you sure you want to exit during a retake session?	When closing the Viz Mosart client during a retake.

Story Recorder Will Not Enable

Story Recorder did not start: the Mosart Logging Service is not responding

Symptoms: the switch shows ON but the status stays SR Off, the item list stays empty for the whole show in the Story Recorder Panel, and a few seconds after enabling, the panel shows:

> *Story Recorder did not start: the Mosart Logging Service is not responding. Check that the service is running on the Mosart server, then switch Story Recorder on again.*

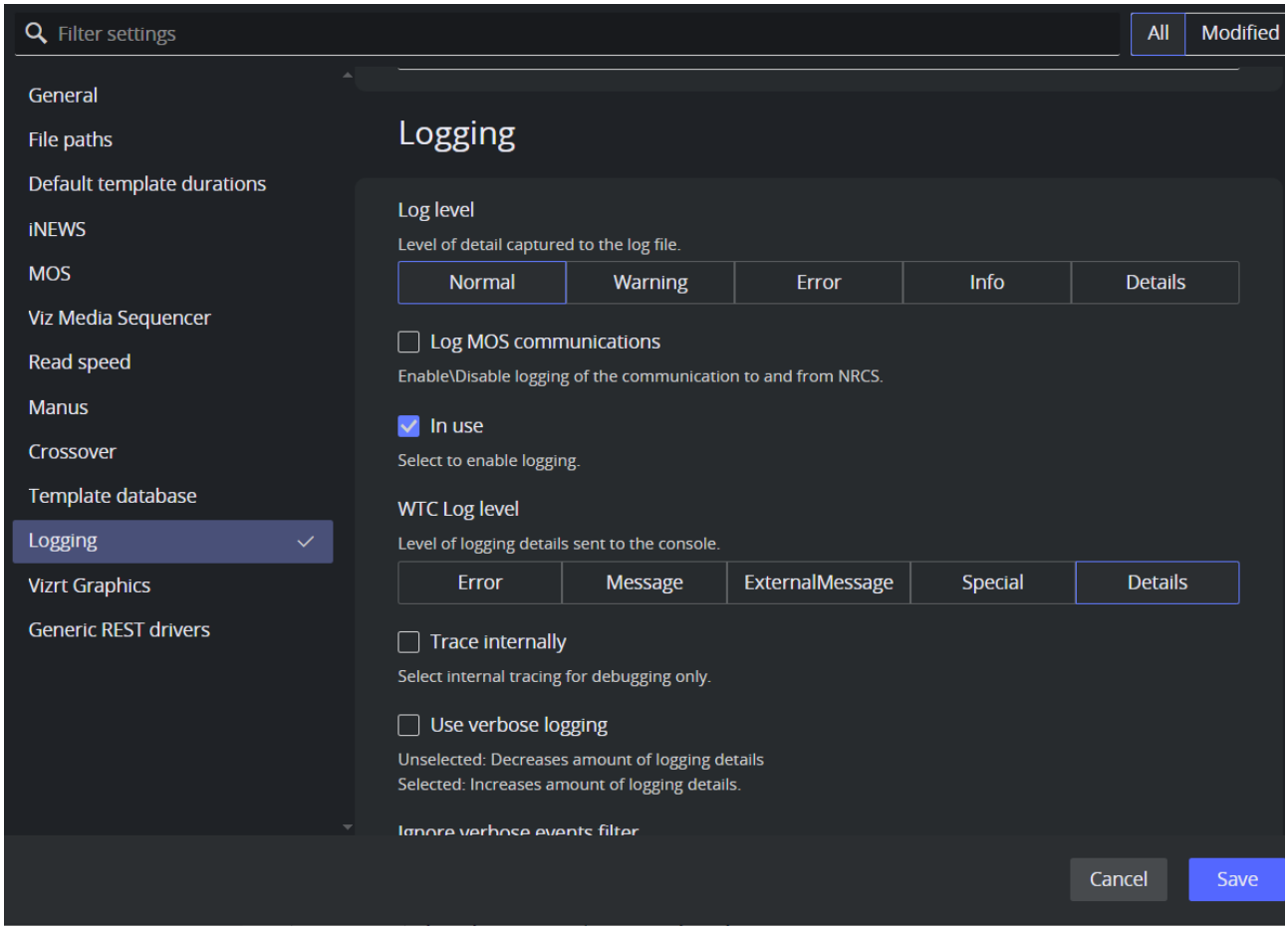


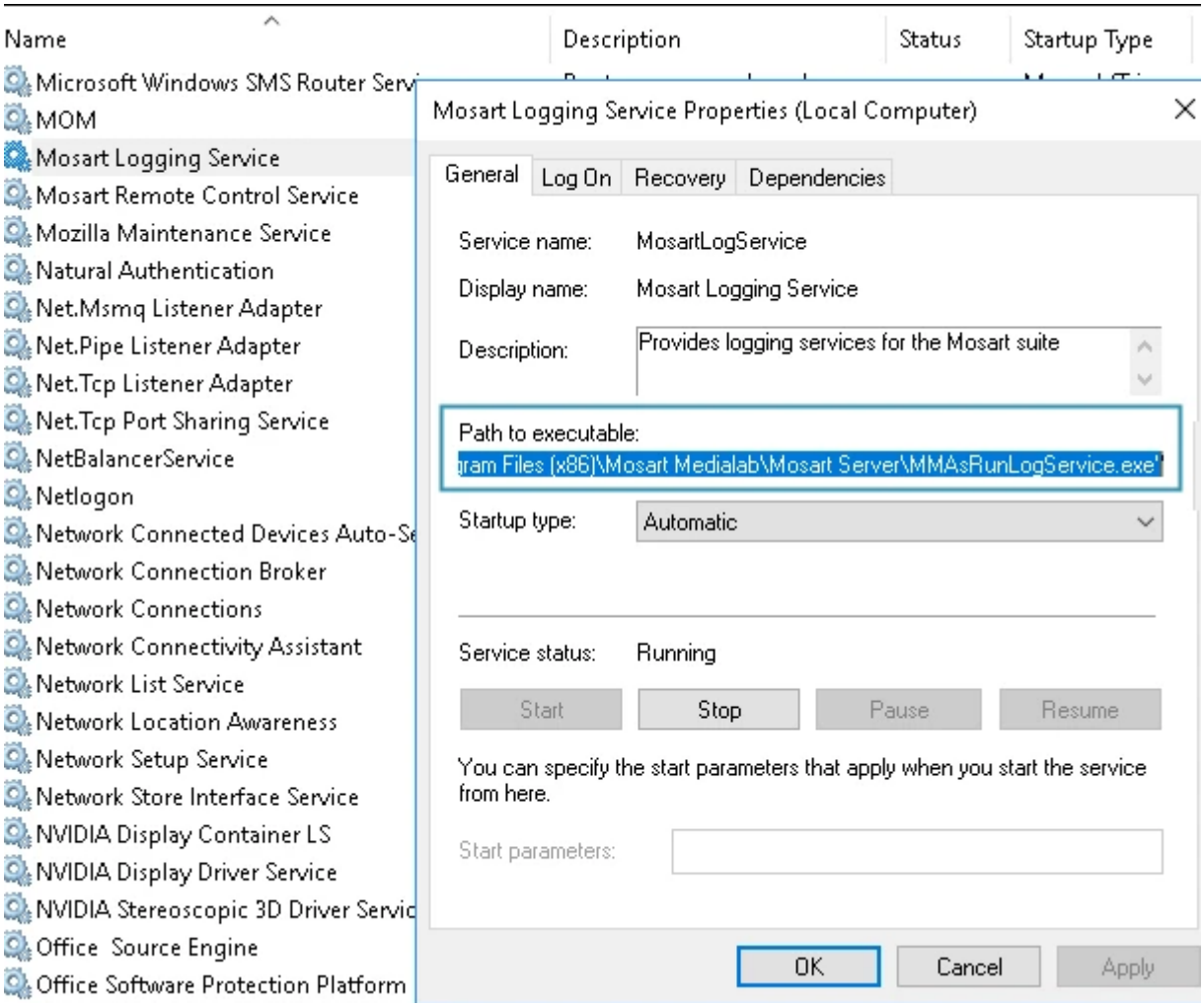
A related message, *Story Recorder did not start, although the Mosart Logging Service is responding*, means the service answered but the Story Recorder adapter failed or was slow to start: switch Story Recorder on again before working through the list below. Note also that when logging is off there will be no Mosart logs to check.

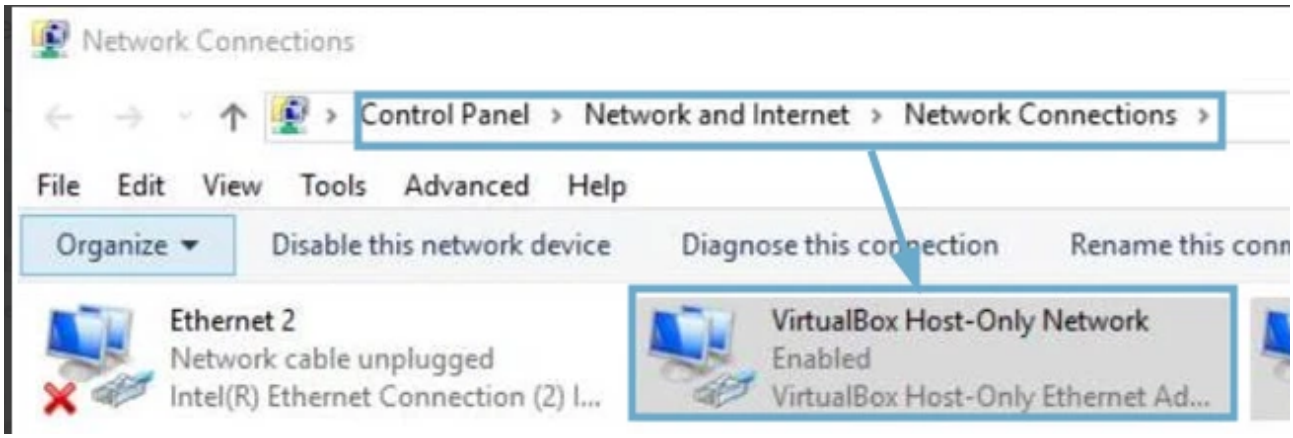
Story Recorder builds its panel entirely from AsRunLog events. Anything that stops those events from being written stops Story Recorder, so work through the checks below in order.

1. Verify in Viz Mosart **Manus Administrator** that logging is enabled: type *settings*, open the **Logging** section, and check that **In use** is selected.
2. **The Logging Service is not running.** Check the Windows service, or run it as a console application.
3. **The Logging Service points at the wrong installation.** This is the most common cause. When the Viz Mosart client, Audio Player, and Server are installed on one machine, the Server must be installed *last*, because the Logging Service has to point at the Server installation folder. If it points at the client or Audio Player folder instead, the Story Recorder adapter is not loaded. Run `MMAsRunLogService.exe` from the *Mosart Server* installation folder as a console application to confirm, then reinstall in the correct order.
4. **A second instance refuses to start.** A Mosart service started twice from the same folder, for example as a Windows service and from a console, does not start a second time. It reports *<application> is already running on this machine, started from <path>. Only one instance can run at a time, as a Windows service or from a console, not both. This one will not start.* The instance already running keeps working, so decide which one you want, stop the other, and start it deliberately.

5. **A virtual network adapter is interfering.** If the client connects to a remote server, make sure no VirtualBox or comparable virtual adapter is installed on the server machine.
6. If the *Viz Mosart Logging Service* is restarted while other applications are already running, those applications stop writing to the AsRunLog and **do not recover on their own**.
   After any restart of the Mosart Logging Service, also **restart AV Automation and the Overlay Graphics Interface**.
   Until they are restarted, events are missing from the AsRunLog and Story Recorder does not work correctly. For example, recording may appear to start OK, but no story items show up in the Story Recorder panel, and graphics encountered in the rundown cannot be backtracked.
7. **Settings are invalid.** Check for the message *Failed to validate the following properties*, correct the listed settings, and restart the Mosart Server Log Service.





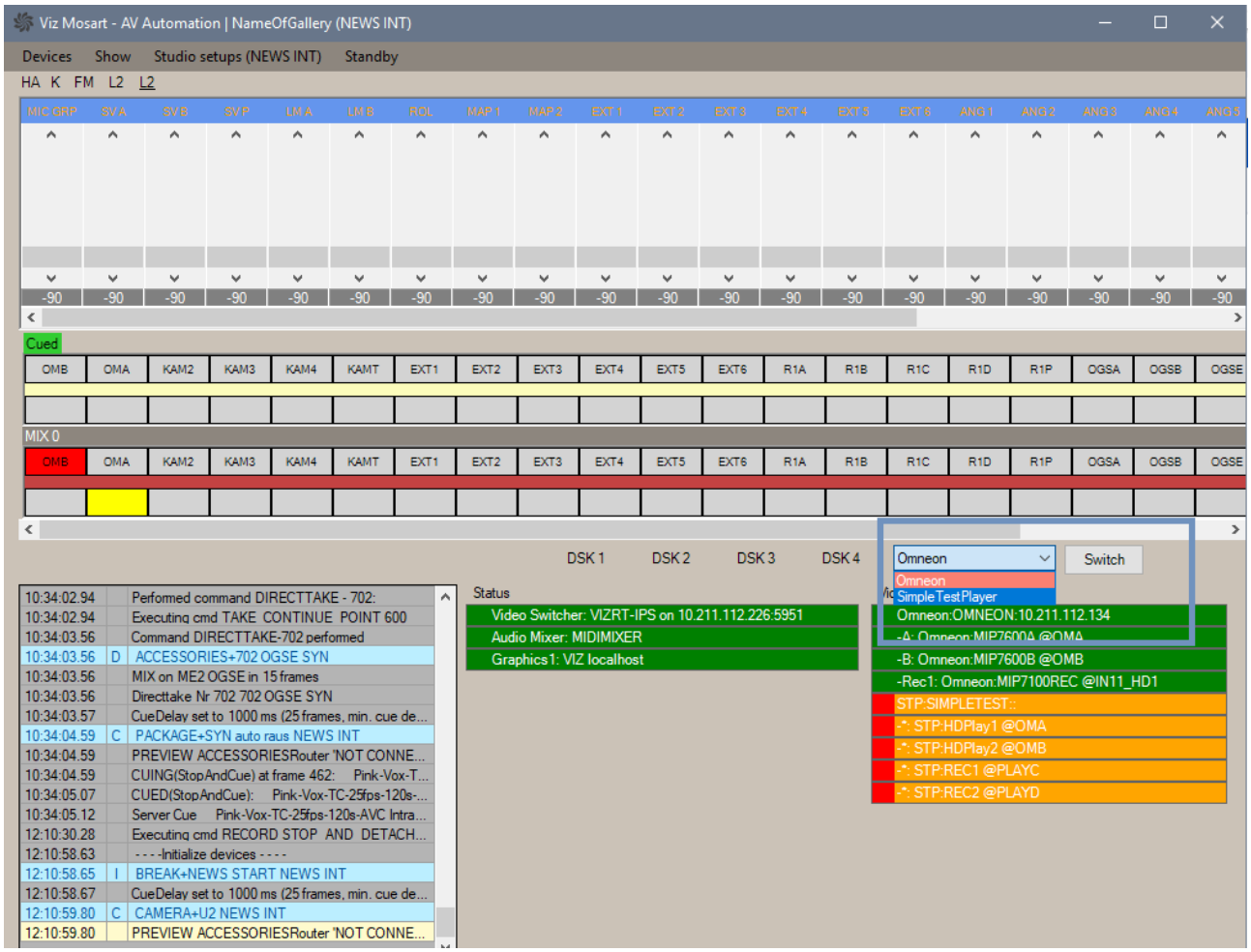


After any restart of the Mosart Logging Service, also **restart AV Automation and the Overlay Graphics Interface**. Without this, the first story item of a show produces no as-run events, and graphics state cannot be restored on a retake.

Recording ports not found

The message *No recorders configured for Story Recorder* means no usable record port was found.

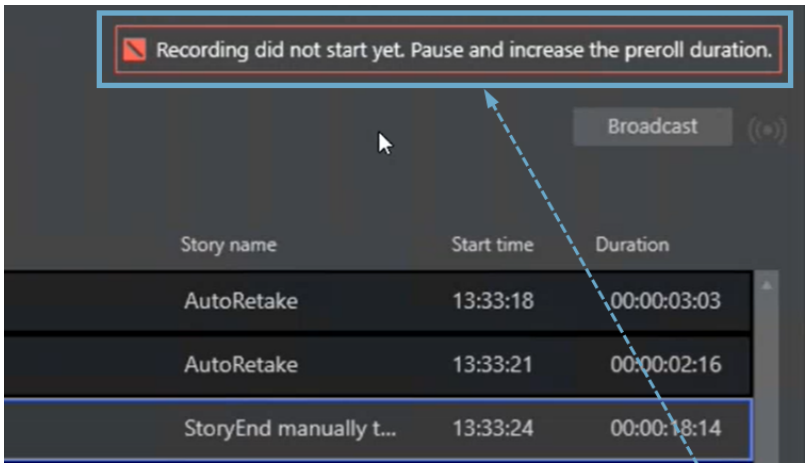
- Check the recorder port configuration in AV Automation, under **Devices > Properties > Video Servers**, and confirm that the port is enabled in the Story Recorder section.
- Add a record port to the current salvo if none is present. Recordings are prepared on the record port of the current salvo, so if the message appears after a salvo change, check that the new salvo has a record port.
- After changing the port configuration itself, force a reconnect with **Ctrl+Shift+V**, or restart AV Automation.



Recording Does Not Start or Items Are Missing

Recording not starting

If starting a recording produces



the pre-roll time is not long enough for the recording hardware to prepare itself.

- At the top of the Story Recorder panel, increase **Preroll (seconds)** to the hardware start time plus one or two seconds, typically 10 to 15 seconds in total.

> The pre-roll can never be shorter than the re-arm time of the video server.

Story items do not appear in the Story Recorder panel

Story Recorder adds a row when a story item affects the PGM bus of the virtual PP. An item that does not switch a crosspoint on PGM is executed normally but produces no row. This is expected behavior, not a fault. For the full list of what is recognized and what is ignored, see the Show Design for Frame Accuracy and Story Recorder section.

If items that *should* appear are missing:

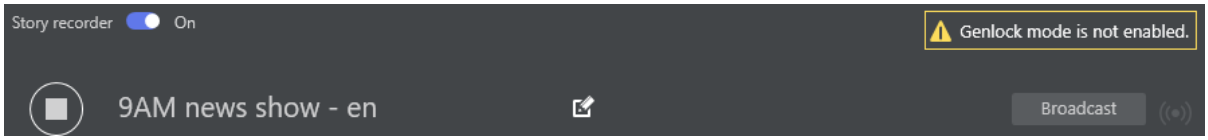
- Confirm that logging is enabled in Viz Mosart.
- Confirm that the Mosart Logging Service is running and points at the Server installation.
- After a Logging Service restart, perform an action in the client such as reloading the rundown, so that the connection is re-established.
- Restart the Overlay Graphics Interface, otherwise graphics events are not logged and graphics cannot be backtracked on a retake.

Timecode and Genlock Problems

These faults usually show up as clips whose content is shifted rather than as a failure to record.

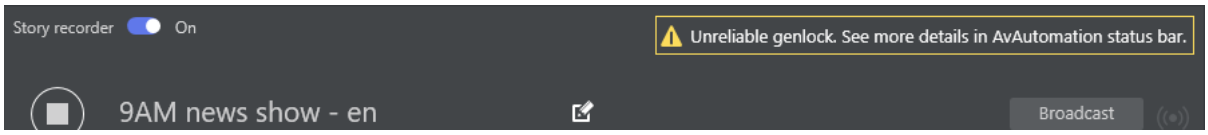
Genlock mode is not enabled

Genlock is not active, so Story Recorder falls back to time of day for its timecodes. Recording still runs, but edit points are not frame accurate. This is acceptable for testing only. For production use, enable genlock and complete calibration. See the Frame Accurate System Operations section.

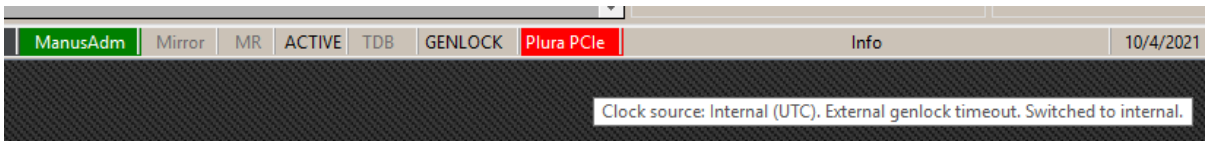


Unreliable genlock

The warning about unreliable genlock means Viz Mosart detected a problem with the clock source and fell back to its internal clock. An occasional occurrence is the built-in safety mechanism working as intended and can be ignored. Frequent or persistent occurrences point to a genlock or timecode signal problem that must be fixed.



The AV Automation status bar will also indicate that there is an issue with the selected clock source:



If the warning is shown infrequently, then the message can be safely ignored and the show can continue as planned.

Viz Mosart has a built-in safety mechanism which takes care of intermittent or lost external genlock and/or timecode signal (refer to section **Genlock and Timecode Reliability** in the topic Frame Accurate System Operations).

If the warning appears frequently or it is persistent, the genlock signal and/or timecode signal have become unreliable and must be fixed.

Time sync issue between the video server and Viz Mosart

Story Recorder compares the start of media reported by the video server against its own clock when recording starts. If the difference exceeds the tolerance, it logs the difference in frames and raises a warning. Recording is not blocked, but mark in and mark out values may be offset.

Copyright © 2026 Vizrt

Situation	Tolerance
External clock in use and timecode synchronized	0.5 seconds
Internal clock, or timecode not synchronized	1.0 second
No clock source available	1.0 second, compared against time of day

The comparison runs once per recording, at record start. It is not repeated during the session, so two clocks that pass the check and then drift apart produce no second warning. This is why both machines should hold time from NTP rather than being set close by hand.
Check that Viz Mosart and the video server use the same time convention. Story Recorder computes each mark as the story timecode minus the start of media reported by the video server, and it uses that start of media exactly as delivered, so a difference in convention becomes a fixed offset. **Use local time** in Manus Settings must match the time reference of the master clock that feeds the video server.

External clock is local time in AV Automation is a separate setting. It describes whether the incoming timecode signal carries local time or UTC, and Viz Mosart converts between the two when they differ. It does not have to match **Use local time**.

> The warning names clock synchronization, but it fires for anything that makes the start of media disagree with wall clock. A frame rate mismatch does it. So does a recorder port that points at the wrong output. If genlock and timecode both check out, look next at The frame rate does not match the video server and Viz Mosart is reading the wrong recorder output.

All clips are offset by the same amount

Every story is out by the same amount, and the offset does not grow through the show. The size of the offset tells you which cause it is.

A few frames

The mark lands slightly before or after the picture it should cut on. For example the EDL says a story starts at `22:07` while the video actually starts at `22:10`: three frames early.

This is equipment latency. Between Viz Mosart telling the switcher to put a source on air and the recorder writing that frame to the file, time passes, and Viz Mosart has to be told how much. It is corrected in **AV Automation > Devices > Properties > Genlock**, in the **Calibration** group.

Adjust **PGM-Recorder**. Add the number of frames the EDL is early, so an EDL three frames early with PGM-Recorder at 0 becomes 3. If the EDL is *late* rather than early, reduce the value by the same amount.

Do not use **Switcher** for this. That value also controls when Viz Mosart sends commands during a live show, so changing it to correct an EDL offset re-times the show itself. **Video Server** is for clip playout and does not affect marks at all.

Click **Apply**. No restart is needed. If a session is running, the new value is used from the next one.

Around a second, or an exact number of hours

Too large for equipment latency, and calibration cannot correct it.

A whole number of hours is a time convention mismatch: the recording carries UTC while Viz Mosart works in local time, or the reverse. See *Time sync issue between the video server and Viz Mosart*, and on TriCaster, *TriCaster start of media is UTC*.

Around a second, with the timecode feed healthy, usually means Viz Mosart is reading the start of media from a different recording than the one it is cutting. See *Viz Mosart is reading the wrong recorder output*.

Wildly wrong, or a negative position

The frame rate in Viz Mosart does not match the recorded output. The error scales with the time of day the recording started, so it is large and looks arbitrary. See *The frame rate does not match the video server*.

The offset is a few frames but genlock is not calibrated at all

If calibration has never been run, do that rather than guessing a single value. See *Frame Accurate System Operations*.

If the offset **grows** through the show, it is not this section. See *Unreliable genlock*.

Only the first story is offset

The request for start of media is asynchronous. If the first story is taken before the video server answers, that story falls back to an estimate from time of day while later stories use the value reported by the server. Allow the session to settle before taking the first story.

The frame rate does not match the video server

Viz Mosart computes each mark as the story timecode minus the start of media, using the frame rate configured in Viz Mosart. On a TriCaster, the start of media arrives over the TriCaster REST API, and Viz Mosart converts it using the frame rate that TriCaster reports for that output. Both halves of the subtraction can therefore be expressed in different units. Nothing in Viz Mosart compares the two frame rates, so a mismatch passes unnoticed.

Symptom. A mismatch does not produce a small offset. The error scales with the time of day at which the recording started, so clips start at a wildly wrong point in the recording, or at a negative position. The export then fails or returns the wrong material.

Fix. Set the Viz Mosart frame rate to the frame rate of the recorded output.

> TriCaster Quick Connect does not import the session frame rate. Someone sets that value by hand in Viz Mosart, so it can disagree with the TriCaster session without any warning. To confirm what TriCaster reports, open `http://{tricaster}/v1/dictionary?key=record_configuration` and read the `frame_rate` attribute on the `<recorder name="OutputN">` row that Viz Mosart records to.

Viz Mosart is reading the wrong recorder output

Viz Mosart matches its recorder port to a TriCaster output row by name, so a port named `Rec1` reads the row named `Output1`. If the program is recorded on a different row than the one Viz Mosart queries, the start of media comes from a different recording. Every clip is then offset by a constant amount, equal to the gap between the moments the two recordings were armed.

Reconfiguring the TriCaster **Setup > Output** tab is a common way for rows to move, so check this after any change there.

Fix. Make the Viz Mosart recorder port name match the TriCaster output row that carries the recorded program.

TriCaster start of media is UTC

On TriCaster the start timecode of a recording comes from the NDI recorder, and it is UTC by default. Enabling LTC in TriCaster overrides that default. When Viz Mosart is configured for local time while the recording start timecode is UTC, every mark carries a whole-hour offset.

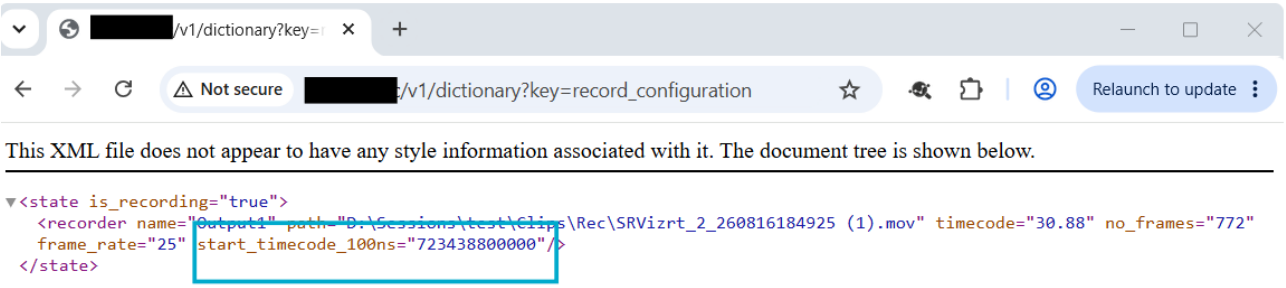
Fix. Enable LTC in TriCaster so the recording carries the house timecode rather than UTC, then set **Use local time** in Manus Settings to match the convention that timecode uses. **External clock is local time** in AV Automation describes the incoming timecode signal and is independent of this; it does not have to match **Use local time**.

TriCaster stops reporting start of media

Viz Mosart asks TriCaster for the start of media of a recording over the REST API, and computes every mark from it. TriCaster can stop answering that request while it otherwise keeps recording normally, and it never answers when it has no timecode source at all.

Symptom. The export job fails, or the story clip is cut at the wrong point in the recording. No start of media for the recording appears in the Mosart log. TriCaster itself looks healthy.

Fix. Restart TriCaster. That clears the case where TriCaster answered earlier in the session and then stopped. Start the recording in TriCaster, wait few seconds and then open this URL in a browser: http://tricaster-server/v1/dictionary?key=record_configuration. The attribute start_timecode_100ns should have a value:



> If TriCaster still reports nothing after a restart, it has no timecode source. Open **Timecode Configuration** in TriCaster, select **LTC**, and set **Source** to **Local (Silence)**. Configure this even when the system does not run in frame accurate mode. See **Preparing TriCaster** in Story Recorder Setup.

Clip Production and Publishing Failures

Mimir: timeout waiting for ingest

Story Recorder waits for the growing recording to be ingested before it asks Mimir to render a story clip. When the wait runs out, no clip is produced for that story.

The log signature. Three lines appear together:

- *Timeout waiting for ingest on recording <id> to render clip for item <id>. Needed frames: <n>*
- *Recording <id> not ready for rendering story clip <id>. Sending the command anyway.*
- *Giving up render story clip <id> from rec <id> after <n> attempts. Last response: 400 BadRequest*

Reading the numbers. `Needed frames` is how much of the recording had to be ingested before the clip could be cut. Divide it by the system frame rate to get that amount as a duration, which is how far into the recording the story ended.

The attempt count separates two different faults. `after 1 attempts` means the ingest wait consumed the whole retry budget, so the render never got a real attempt. A count above one means ingest completed and Mimir rejected the render itself, which is a different problem and is not solved by changing the retry settings.

How the retry settings behave. The Mimir retry settings do not work the way their names suggest, so read this before changing them.

- **Retry start delay** elapses first and is not counted in the budget.
- **Retry total duration** is a single budget shared by the ingest wait and the render retries. A long ingest wait leaves nothing for the retries.
- **Retry interval** is the gap between ingest checks, and then the gap between render retries.
- A hard ceiling of 15 render attempts applies regardless of the time budget.
- Only a validation error meaning the media is too short is retried. Other errors stop the sequence immediately, whatever the settings say.

> **The quickest way to tell the causes apart.** During a show, open the recording item in Mimir and watch its duration. A duration that is not growing points at the recorder or its format. A duration that grows but stays behind the show points at ingest throughput.

Check the causes in this order.

1. **The recorder is not using a low latency wrapper.** On a Harmonic Spectrum, the record port **Media Wrapper Format** must be *MXF OP1a (Standard, Low Latency)*. Plain *MXF OP1a (Standard)* delays exposure of the growing file, so ingest cannot keep up. This was the confirmed cause in a real customer case.
2. **The recorder is not using an I-frame only format.** Long GOP intermediate recordings are not supported.
3. **Ingest throughput is behind the recording.** Mimir is taking in the growing file more slowly than the video server writes it. Fixing that is an ingest-side performance question rather than a Viz Mosart setting.
4. **The retry budget is too short for the story length.** Convert `Needed frames` to a duration and compare it against **Retry total duration**.
5. **The Kelda config ID is wrong.** Mimir then uses the wrong ingest configuration. Confirm that the ID matches the ingest configuration in the Mimir tenant.
6. **The path between the video server storage and Mimir is slow or unreachable.** Check the network route and the storage that Mimir reads from.

Those six cover the Viz Mosart and recorder side. Causes on the Mimir and Kelda side are not limited to the ones listed, because the ingest and render steps run there rather than in Viz Mosart. Viz Mosart can only report that the recording did not become available in time.

What to check on the Mimir side. Go through these with whoever administers the Mimir tenant.

- Whether the recording item exists in Mimir at all. Viz Mosart creates a placeholder item first, and reports *Failed to create recording placeholder item in Mimir* if that step fails. Viz Mosart then polls that item to watch ingest progress, so a missing or wrong item means the wait can never succeed.
- Whether Kelda is ingesting that item, and whether the ingest is progressing, stalled, or never started.
- Whether the **Kelda config ID** in the Story Recorder settings matches an ingest configuration that exists and is active in the Mimir tenant.
- Whether Mimir and Kelda can reach the storage where the recording is being written. The render request refers to the recording by path, so the ingest side needs access to that location.
- Whether the **Preset ID** matches a valid render job preset in the Mimir system settings.
- Whether the Mimir tenant has capacity, or whether ingest and render jobs are queueing behind other work.

If the recording item is not progressing in Mimir while the recorder is confirmed to be writing a growing file, the investigation belongs with the Mimir and Kelda side rather than with Viz Mosart. Hand over the three log lines, the recording item identifier, the anchor clip identifier, the Kelda config ID, the preset ID, and the time the recording started.

Mimir: other failures

- *Failed to create recording placeholder item in Mimir.* Check the Generic REST device, and that Mimir is reachable.
- If a render fails when transcription is enabled, check that the language code and transcription service pair matches a transcription configuration in the Mimir tenant settings. An unmatched pair fails the render.
- Invalid JSON in any of the optional request templates is ignored with a warning and the defaults are used, so it never fails the call on its own.

TriCaster export

If an export does not appear, check the export preset names, which are separated by a vertical bar, and the credentials in the video server connection string. For story clip production, the delay before the export job is submitted gives TriCaster time to finalize the growing recording. Increase it if exports fail on long stories.

Vantage and EDL output

- **Nothing arrives at the transcoder.** Check that the EDL watch folder is reachable and that the Mosart Server Logging Service account can write to it. The local EDL folder and the watch folder must be different.
- **Duplicate jobs.** Every press of **Deliver** sends a new task and creates a new take. The counter is cumulative and never decreases. Cancel unwanted jobs in the transcoder job queue.

Snippet review clips

If no review snippet appears after a retake, check that Viz Coder is installed, that the snippet folder is configured and writable, and that a default player for `.mxf` files exists on the client machine. Snippet failures do not affect the retake itself.

Pause and Retake Errors

Pause is ignored

The warning *Pause ignored: retakes are switched off in the EDL output settings* means **Enable retakes** is off, so the timeline cannot be paused for a retake. The retake, preroll, and pause controls are greyed while it is off. If the workflow uses pause and retake, switch **Enable retakes** on in the EDL output settings and save while Story Recorder is off. See Show Clip Production Setup.

Invalid retake position

If you get the error notification *Retake position no longer valid. Please set retake point* this means that you restarted the show after a pause, but the *selected story item* is not *the retake point* (the story item with the **Set Retake** button):

To fix, do one of the following:

1. Select the *current retake point*, if this has changed by mistake, or
2. Make the *selected story item* the new retake point by clicking the row's **Set Retake** button.

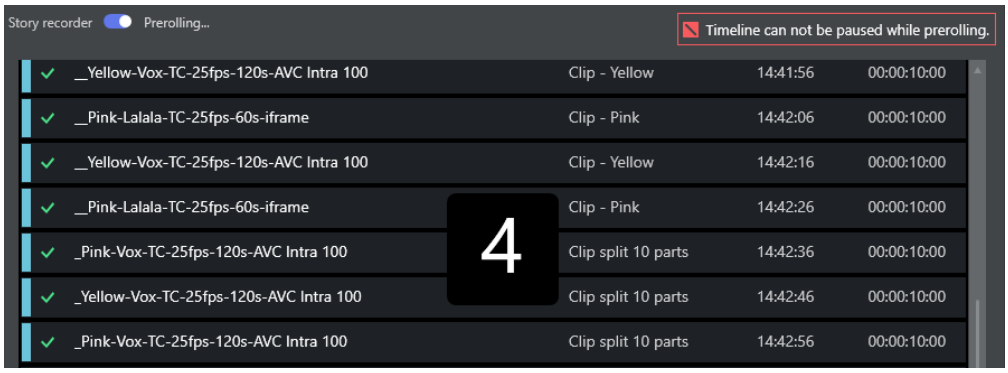
> ⚠ If you choose (2), remember that all the recorded story items after the retake point, including the story item set as retake point, must be re-done once the show is restarted after pause (see Working with Story Recorder).

Retake point replaced by an earlier retake

The warning *The retake point is no longer valid: the take it pointed to was replaced by an earlier retake. Select another retake point* means the take that the selected retake point referred to no longer exists, because an earlier retake superseded it. Nothing is removed: the recorded items and their EDL entries are kept. Select another retake point and continue.

Error when pausing the story

If you get the error notification *Timeline can not be paused while pre-rolling*, this means you are trying to pause while the system is in pre-roll mode. The pre-roll process can not be interrupted.



If this action was done intentionally, let the pre-roll time out and auto-take the item to be retaken, and then pause again.

Retake requested too early

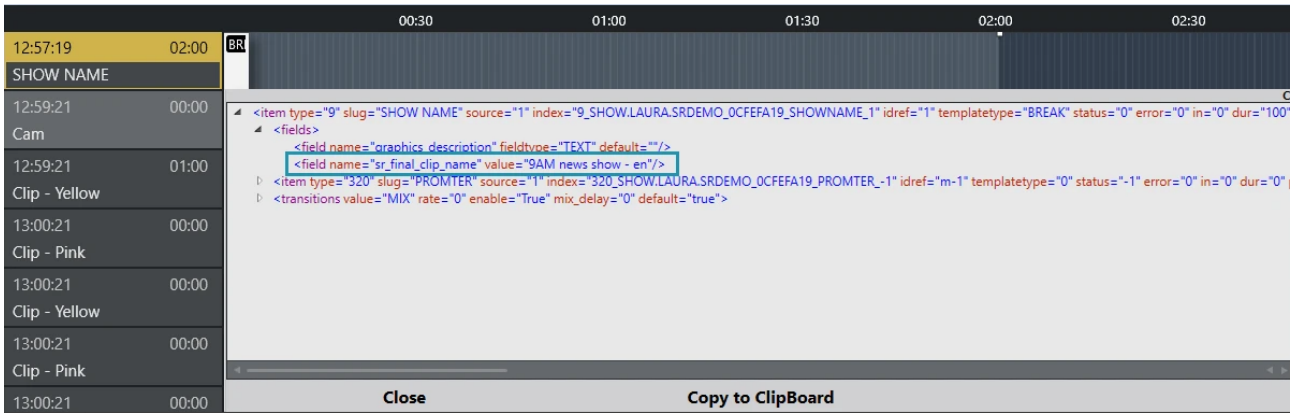
The messages *Waiting for recorder to pause* and *The recording has not yet started* both mean the session is not yet in a state where a retake can be set. Wait until the panel shows Paused, then try again.

Naming of the Final Clip

If a final clip name supplied by the NRCS does not appear in the Story Recorder panel:

- The name is read from the *first* story in the NRCS rundown. Check the status of the first story. If it has been set to *float* and is not displayed in Viz Mosart, add the name to whichever story is now first.
- Check whether another story has been set as next, which changes which story is treated as first.
- A name changed in the NRCS during a session takes effect only at the next session. Disable Story Recorder mode and enable it again.
- To inspect what the NRCS actually sent, hold **Ctrl+Shift+Alt** and double click the first story item to view the MOS XML.

Copyright © 2026 Vizrt



Housekeeping and Disk Space

Recorded files are always large. Without a purging regime in place, the recording volume will fill.

- **Intermediate recordings** can be purged automatically using the recording clip name prefix together with the retention period. The task runs every time the Mosart Server Log Service starts or restarts. The service account needs read, write, and delete rights on the recorder port folders. A retention value of zero disables purging.
- **EDL folders are not managed by Viz Mosart.** Neither the local EDL folder nor the watch folder is cleaned up automatically, so both need an occasional manual purge by the local system administrator.
- **The final show clip folder** is controlled by the transcoder, not by Viz Mosart.
- All settings, such as pre-roll duration and file locations, are preserved between sessions.

Limitations

- A separate, dedicated template set is required for shows that use Story Recorder. These templates cannot be used when Viz Mosart runs in non-Story Recorder mode.
- You can only retake from the *beginning* of a primary story item. A single item cannot be replaced in isolation: everything from the retake point onward must be redone.
- The following story elements and secondary items give unpredictable results around the cut point and must be carefully reviewed: accessories, manual commands, running animated graphics, and state variant behavior.
- Because Viz Mosart runs in *Frame Accurate mode* rather than *Standard mode* during a Story Recorder show, not every feature from Standard mode is available. See the Show Design for Frame Accuracy and Story Recorder section for the full matrix.
- Reloading a rundown ends the session by default, publishing what has been recorded, and Story Recorder must then be enabled again. This behavior is controlled by the setting that disables Story Recorder on rundown reload. When that setting is switched off, the session moves to Paused instead and can be resumed.
- Reloading a rundown into rehearsal while recording pauses the recording, and no clips are produced.

Copyright © 2026 Vizrt

Tip: Use the *snippet player* to verify the cut point (see Working with Story Recorder).

Collecting Data for Vizrt Support

Timing problems cannot be diagnosed from a description alone. Collect the following before contacting Vizrt Support.

Files

Item	Where	Why it is needed
Viz Mosart technical logs	`C:\MMLogs\`	Crosspoint events, start of media, cue and play commands, with timecodes.
EDL files	The configured EDL folder, by default `C:\MMLogs\EDL`	The mark in, mark out, and start of media values actually produced.
Genlock logs	`C:\MMLogs\Genlock`	Genlock behavior for the session.
Clock source statistics	The configured statistics folder, by default under `C:\MMLogs\GenLockStats\`	Whether the external clock and the frame clock were reliable. See the Frame Accurate System Operations section for how to gather these.
The source recording	Video server clip storage	Lets Support compare the marks against the actual media.
The rendered clip	Mimir, TriCaster, or the export location	Shows what the viewer would have seen.
Configuration files	`avconfig.xml` and the channel templates file from the working folder, the AV Automation settings and the settings from C:\ProgramData\Mosart Medialab\ConfigurationFiles.	Frame rate, video server type, genlock and calibration values, transition types.

Settings to report

Frame rate, video server type, whether genlock is enabled, the clock source in use, and the four calibration latency values from the AV Automation **Genlock** tab.

Questions to answer

- Which story and which template were on air when the problem occurred?
- What action triggered it?
- What was expected, and what happened instead?
- Is there a screenshot or a short video?
- Is it reproducible, and if so how often?

How the offset behaves narrows the cause quickly. A constant offset across all stories points at calibration or a missing timecode signal. Random offsets point at a synchronization problem. An offset that grows through the show points at a clock mismatch between Viz Mosart and the video server.

Copyright © 2026 Vizrt

13 Viz Mosart Template Database

With the Viz Mosart Template Database (TDB), template information can be shared among the following clients:

- Between galleries automated with Viz Mosart (where templates are shared across galleries and Mosart servers).
- Between a Mosart main and backup server, in a redundancy setup.
- To NRCS clients using the Mosart Web Apps NRCS Plugin.
- To NRCS clients using Mosart ActiveX or a custom ActiveX.

The following Template Database topics are covered:

- Architecture
- Template Properties
- Information Stored in the Template Database
- Prerequisites for Connecting to TDB

13.1 Architecture

- The template database is stored in a MySQL relational database (or a valid MySQL equivalent such as MariaDb).
- The database may be installed separately for the Mosart servers or as a part (schema) of an in-house or cloud database, or even be virtualized.
- The only main requirement that the database must be MySQL compliant.

13.2 Template Properties

A Mosart template as stored in the template database has the following properties:

Property	Description
Gallery	The Mosart gallery naming where this template is implemented, and 'owner' of the templates. **Note:** A special *global* gallery is used for TDB templates that are shared, replicated and available between *all* galleries. This dedicated, virtual gallery for a global template set, is by default named *SHARED*.

Property	Description
Template set	The Viz Mosart template set to which a template belongs. Can be *shared* or *local*. **Note:** A local template set is **gallery-specific,** meaning it is 'owned' by the corresponding gallery and is independent and not visible to any other galleries. **Note:** The special Mosart template set **Directtakes** must be defined as a *shared* template set.
Template type	The Mosart template type. For example, CAMERA, CLIP, DVE.
Template variant	The Viz Mosart template variant. For example a specific camera is identified with a template named *CAMERA+3*.
Template description	Common attributes for the template which is shared among all galleries. The template description also contains a list of input fields normally known as *newsroomtags*.
Template implementation	Gallery-specific information used to control systems using Viz Mosart for automation. This is the main part of the template and has to be implemented in every gallery.

13.3 Information Stored in the Template Database

Details from the template database are stored in two Mosart server applications:

- **AV Automation**
  Stores templates and the file AvConfig.xml.
- **Manus Administrator**
  Stores NRCS aliases as configured in the **Newsroom settings** dialog.

No other information in `newsroomsettings.xml` is stored in the database.

13.4 Prerequisites for Connecting to TDB

Before connecting to the template database the first time, make a backup of `channeltemplates.xml`, `AvConfig.xml` and `newsroomsettings.xml`

- Both AV Automation and Manus Administrator extract their information from the database and store the information locally in `channeltemplates.xml`, `AvConfig.xml` and `newsroomsettings.xml` respectively.
  Any existing files are overwritten.

This section continues with:

- Creating and Upgrading Mosart templatedb
- Backup and Recovery
- Database Maintenance
- Viz Mosart Template Database Specification
- Configuring the Viz Mosart Server
- Installing the Viz Mosart Database

13.5 Creating and Upgrading Mosart templatedb

The Viz Mosart Template Database uses a schema called *mosarttemplatedb*.

Prerequisties

- Before continuing, ensure Viz Mosart has been installed. Refer to section Installation, and the latest Release Notes.
  A Viz Mosart installation includes an essential database creation batch file *RunDbScripts.cmd* written to: *%ProgramFiles%\Mosart Medialab\Mosart Server\BatchFiles.*
- An installed WampServer (refer to section Installing WampServer).

This section contains the following topics:

- Upgrading MySQL 5.x to MySQL 8.x
- Verifying the WampServer
- Running Database Scripts
- Running Scripts Automatically
- Running Scripts Manually

13.5.1 Upgrading MySQL 5.x to MySQL 8.x

Migrating the Mosart templatedb

In connection with the migration of the template database to the new version of the database server, it will be necessary to transfer all data from the old, following the recommendations in Template Database section Backup and Recovery.

To Migrate the Template Data

1. Transfer tables and data to the new server.
2. Run the command below.

```
ALTER USER ‘username’@’*’ IDENTIFIED WITH ‘mysql_native_password’ BY
‘password’;
```

replacing `‘username’` and `‘password’` placeholders to their actual values.

13.5.2 Verifying the WampServer

To Verify WampServer is Running

1. Check the WampServer icon is present in the system tray.
   a. If not, start WampServer from the **Start** menu.
   b. It should then appear *green* in the System tray.
2. If the WampServer icon is not green, click the icon and select **Start All Services**.

13.5.3 Running Database Scripts

The Viz Mosart Database can be created automatically using the *Db00Create.sql* script.

Location of Script

Navigate to the following locations, for x86 and x64 machines respectively:

- `%ProgramFiles%\Mosart Medialab\Mosart Server\Sql`
- `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\Sql`

There are also several *upgrade* scripts:

- Db01Change….sql
- Db02Change….sql
- and so on

Database changes are implemented as upgrade scripts rather than as new versions of the create script. This is to avoid destroying existing data.

- Scripts may also be run manually. It is recommended to run scripts automatically.
- If problems arise when running scripts automatically, more information on manually running scripts can be found later in this section.

13.5.4 Running Scripts Automatically

Scripts run from a batch file, *RunDbScripts.cmd*.

Location of Scripts

Navigate to the following locations, for x86 and x64 machines respectively:

- *%ProgramFiles%\Mosart Medialab\Mosart Server\BatchFiles*
- *%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\BatchFiles*

The file refers to the MySQL command line tool mysql and an .ini file used by that tool.

- The .ini file is installed into the same folder as the batch file, but contents should be verified and amended as needed.
- The reference to the tool itself should also be verified.

To Run the Database Scripts

1. Edit, do not open, *RunDbScripts.cmd*. File contents should be similar to the following:

```
REM This file MUST be saved with Encoding = ANSI
for %%f in (<file list>) do C:\wamp\bin\mysql\mysql5.5.8\bin\mysql
--defaults-extra-file=mysql.ini < %%f
pause
```

2. Check if the stated folder (in this example, *C:\wamp\bin\mysql\mysql5.5.8\bin*) exists and contains *mysql.exe*. If so, continue with step 5. If not, continue to step 3.
3. Locate *mysql.exe* and change the contents of *RunDbScripts.cmd* accordingly.

4. Save *RunDbScripts.cmd* with ANSI encoding.
   For example, in Notepad, choose **File > Save As**, then **Encoding: ANSI**.
5. Edit, do not open, *mysql.ini*. File contents should be similar to the following:

```
[mysql] host=localhost database=mosarttemplatedb user=root password=
```

6. Verify entries in the file and change if necessary:
   - `host=localhost` : If MySql is running on another PC, replace localhost by the name or IP address of this other PC.
   - `database= mosarttemplatedb` : Should not be changed.
   - `user=root` : This is the default admin user. If you want to run the scripts with another user, replace with the name of a user with sufficient privileges.
   - `password=` : The default root password is empty. Replace if you have changed the root password or have replaced root.
7. Save the file if you have changed it.

13.5.5 Running Scripts Manually

To Run the Database Scripts Manually

1. Start phpMyAdmin by **left**-clicking the WampServer icon, and selecting the appropriate phpMyAdmin under PhpMyAdmin.
2. Create the database, *mosarttemplatedb*, if not already done.
3. Upgrade the database using the scripts from the appropriate location:
   - *%ProgramFiles%\Mosart Medialab\Mosart Server\Sql*
   - *%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\Sql*

13.6 Backup and Recovery

The Viz Mosart Template Database is stored in a MySQL database and any backup and recovering strategies offered by the database vendor is valid.
In addition, most MySQL clients offers the possibility to backup and restore the entire database from a single .sql file.

In this section, the MySQL command line tools *mysqldump* and *mysql*are used for backup and restoring respectively.

> **Tip:** *mysql.exe* and *mysqldump.exe* are part of MySQL Server, MySQL Workbench or Wampserver installations.

- Working with the Server Database
  - Database Backup
  - Database Restore
- Working with Database Clients
  - Backup and Restore from a MySQL Client

13.6.1 Working with the Server Database

Database Backup

To backup using mysqldump

- The command above creates a single .sql file containing the entire Viz Mosart Template Database.

```
mysqldump [--host=hostname] --user=username --password=password --add-drop-database
mosarttemplatedb > backupfile.sql
```

Database Restore

To Restore using mysql

- The command below restores the Viz Mosart Template Database from a single .sql file as created using the *mysqldump* command shown in the former section.

```
mysql [--host=hostname] --user=username --password=password mosarttemplatedb <
backupfile.sql
```

Copyright © 2026 Vizrt

13.6.2 Working with Database Clients

Backup and Restore from a MySQL Client

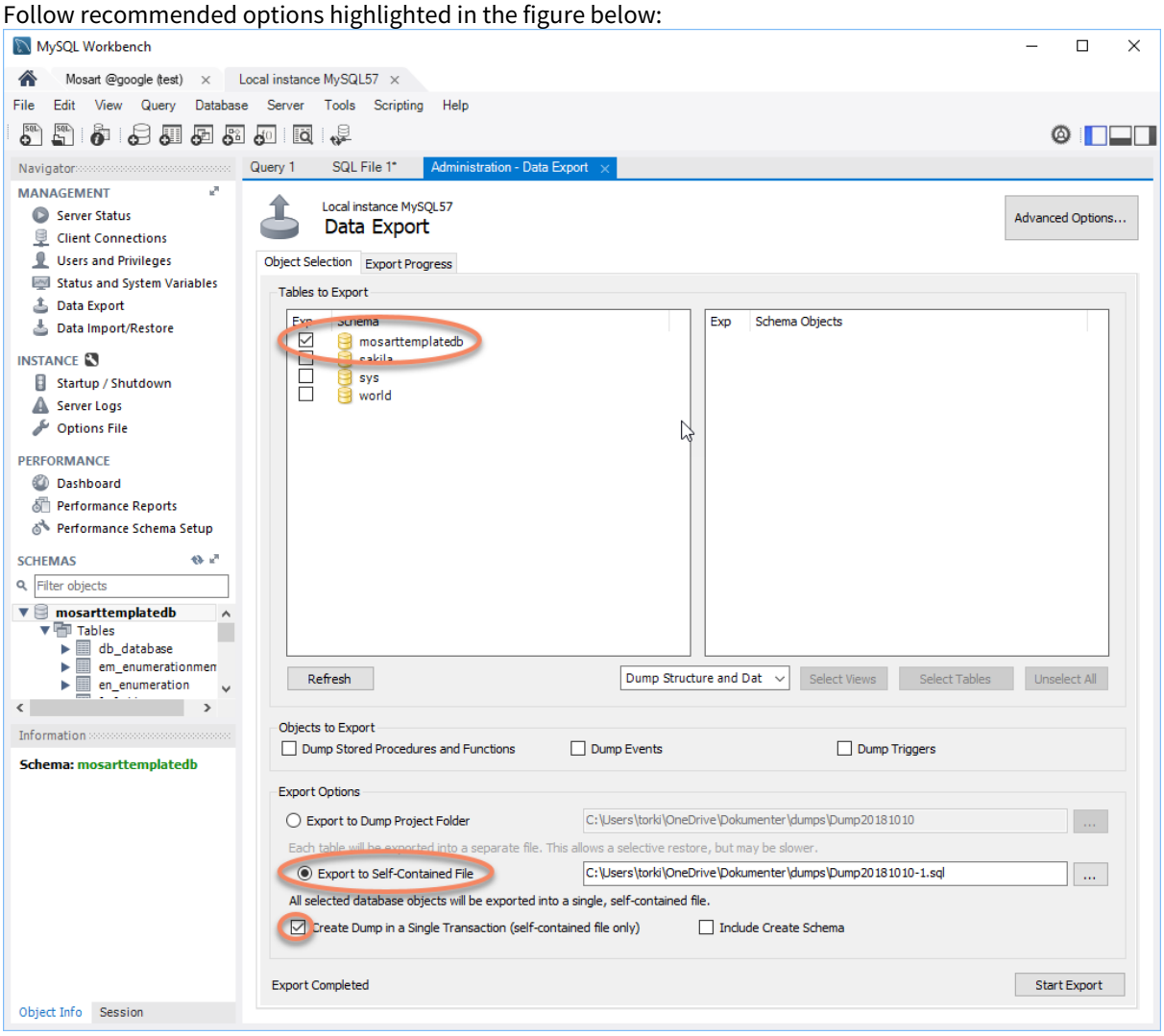
Backup and restore operations using the *mysqldump* and *mysql* tools are also available in all recommended MySQL client applications.

When using a client application ensure the following:

- Database schema = mosarttemplatedb.
- Export to a *single* file. The entire database must be saved to a single file.
- Include *drop* and *create* functions. This ensures that the database is restored as stored on the file.

Example: Backup from MySQL Workbench

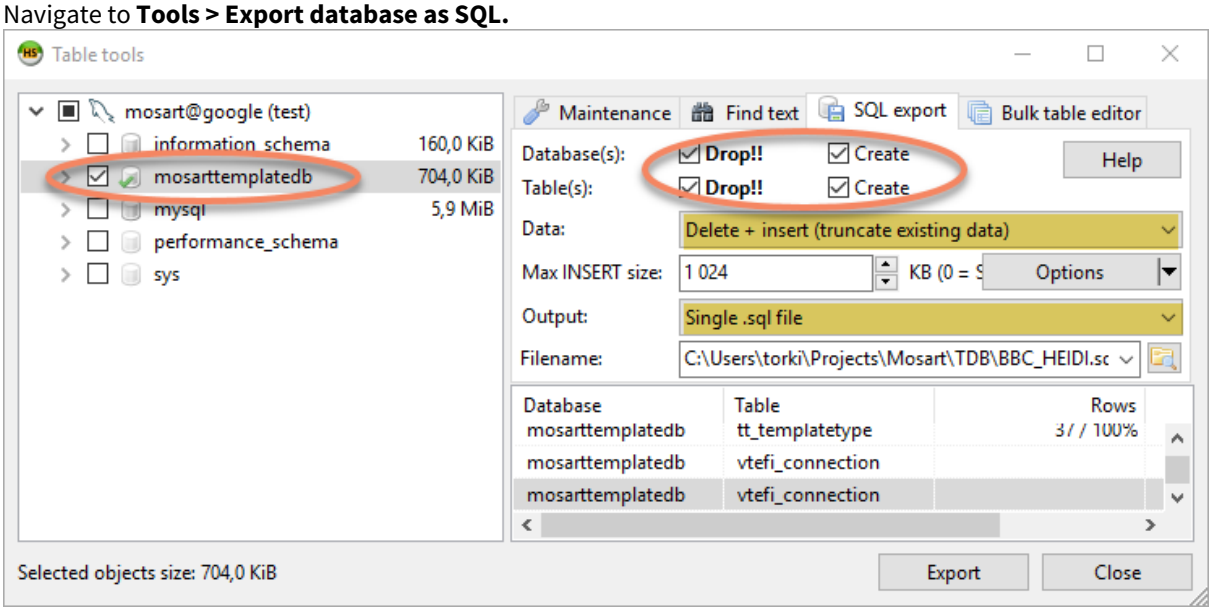
1. In MySQL Workbench navigate to **Server > Data Export > Export**
   This opens the Data Export dialog menu.
2. Follow recommended options highlighted in the figure below:



3. Restore from a single .sql file in MySQL Workbench.
   Navigate to **Server > Data Import**.
4. Select the .sql file as created during backup.

Example: Backup from HeidiSQL

1. Navigate to **Tools > Export database as SQL.**



2. Ensure the following:
   a. For both **Database(s)** and **Table(s)**, *Drop* and *Create* are selected
   b. **Data**: is set to *Delete + insert (truncate existing data)*
   c. **Output**: is set to *Single .sql file*.
3. Navigate to **File > Run SQL file**
   and restore from a single .sql file in HeidiSQL.

13.7 Database Maintenance

After initialization, you perform most changes to the Viz Mosart Template Database in AV Automation and the Template Editor.

These changes may include upgrading the database, adding and removing galleries, adding and removing template sets and adding, removing and editing templates.

This section presents the following topics:

- Upgrading the Database
  - Upgrade History
- Viz Mosart Template Editor
- TemplateSetEditor
  - Configuration of TemplateSetEditor
  - Configuring the Required Template Database
  - Configuring an External Tool for Displaying Differences
  - Configuring an External Editor for XML Content in a Single Template
  - TemplateSetEditor Commands
  - Compare [set1] [set2] [type] [template]

13.7.1 Upgrading the Database

AV Automation will automatically upgrade the database. All upgrades and current database version are stored in the *db_database* table.

> **Note:** Automatic upgrades shall never delete or modify any content in the database. The database should be kept backward compatible at all times.
> Upgrades normally add new functionality to the database

Upgrade History

The following table summarizes the upgrade history of the Viz Mosart Template Database:

Version	Date	Version	Description
0.1.0	01.06.2012	3.2	Initial version.
1.0.0	01.12.2012	3.2	First version of automatic upgrade. New table: db_database.
1.0.1	01.10.2013	3.2	Added support for sequences. New table: sq_sequence.
1.0.2	07.05.2013	3.3	Added templateset properties. New table: tsp_templatesetproperties.

Copyright © 2026 Vizrt

Version	Date	Version	Description
1.0.3	27.06.2013	3.3	Verified default transitions: Added initial content to the tr_transition table.
1.0.4	05.09.2013	3.5	Added order field in tdfi_connection table. New column: tdfi_order.
1.0.5	26.08.2014	3.5	Added scope to ta_templatetypealias table. New column: ta_scope.
1.1.0	17.01.2016	4.0	Added support for hierarchical template sets. New columns in ts_templateset table: ts_basedon, ts_hidefromuser.

13.7.2 Viz Mosart Template Editor

The Viz Mosart Template Editor is normally used to add, delete and modify Viz Mosart templates. When attached to a Mosart Template database every save operation pushes the changes incrementally to the database.

Use the Template Editor for the following changes:

- Adding and removing template sets. Note when adding a new template set it is possible to choose whether the template set shall be shared or gallery specific (local). It is not possible to change this afterwards.
- Adding, removing and editing templates. This is the normal use case of the template editor.

13.7.3 TemplateSetEditor

TemplateSetEditor is a console application that is part of the Mosart Test Suite. This application may be useful for simple maintenance and is currently the only way to remove galleries from the database.
The TemplateSetEditor was made as part of adding hierarchical templates to the database.

Info: As of Viz Mosart 4.0, the TemplateSetEditor is a standard component.

Configuration of TemplateSetEditor

Tip: Configuration of TemplateSetEditor is done in the application configuration file, TemplateSetEditor.exe.config

userSettings

Within the application configuration file locate the userSettings sections where you can:

- Select an external tool for showing differences. Used by the **compare** function
- Select an external tool for viewing single templates or template sets in XML. Used by the **edit** function.
- Select the template database to use with corresponding credentials (connectionstring).

- Select template respository (future). Only *C:\channeltemplates.xml* is currently supported.

Configuring the Required Template Database

You can register three different databases to connect to.

1. Enter the respective connectionstrings using the properties *Database1*, *Database2* and *Database3*.
2. Select which database to use by assigning the desired server (hostname) to the *Database* property.

Example

- *Database1*: gallery=StudioE;server=localhost;User Id=user;password=password;database=mosarttemplatedb
- *Database2*: gallery=StudioE;server=146.148.9.188;User Id=user;password=password;database=mosarttemplatedb
- *Database3*: <empty> (not used)

Setting the property Database to localhost makes use of the connectionstring given in the *Database1* property, setting the *Database* property to 146.148.9.188 makes use of the connectionstring given in *Database2.*

Configuring an External Tool for Displaying Differences

You can reveal differences to template sets or templates by comparing their corresponding XML representations using the **compare** function. Register up to three different tools for comparison using the properties *DiffTool1*, *DiffTool2* and *DiffTool3*.
Use {0} and {1} as placeholders for the two different contents to compare. Then select which tool to use by setting the property *DiffTool* to the name of the desired tool.

Example

Tool	Code Example
DiffTool1: Visual Studio Code	Name=VSCode;Program=C:\Program Files\Microsoft VS Code\Code.exe;Args=--diff {0} {1}
DiffTool2: Beyond Compare	Name=BComp;Program=C:\Program Files\Beyond Compare 4\BCompare.exe;Args={0} {1}
DiffTool3: ExamXml	Name=ExamXml;Program=C:\Program Files\ExamXMLPro\examxmlpro.exe;Args=--diff {0} {1}

Setting the property *DiffTool* to BComp makes use of Beyond Compare as the external tool to use for the comparison.

Configuring an External Editor for XML Content in a Single Template

You can edit or view the XML content of a single template.

Register up to three different tools for editing xml using the properties *EditTool1*, *EditTool2* and *EditTool3*.
Use {0} as placeholder for the content to edit. Then select which tool to use by setting the property *EditTool* to the name of the desired tool.

Examples

Tool	Code Example	Notes
EditTool1: Visual Studio Code	Name=VSCode;Program=C:\Program Files\Microsoft VS Code\Code.exe;Args={0}	Uses Visual Studio Code as the an external editing tool
EditTool2: XmlMarker	Name=XmlMarker;Program=C:\Program Files (x86)\XML Marker\xmlmarker.exe;Args={0}	
EditTool3: Notepad++	Name=Notepad++;Program=C:\Program Files\Notepad++\notepad++.exe;Args={0}	

TemplateSetEditor Commands

Below is a list of commands available in the TemplateSetEditor:

```
TemplateSetEditor
/?                Outputs this help message
Clear             Clears the console window
CleanUp           Clean up templates, removes depricated stuff etc.
Compare           [set] [baseset] [type] [template] Compares template sets or a single
template
Copy              [set] [newset] Makes a copy of set with name equals newset
Edit              [set] [type] [template] Opens a single template in an external xml
editor
End               Terminates Mosart Template Set Editor
Equals            [set] [baseset] [type] [regex] Performs equality test on templateset
or set of templates
Exit              Terminates Mosart Template Set Editor
Flatten           [set] [type] [template] Flattens a templateset or a single template
Gallery           [gallery|add|select|delete|rename|list] [gallery] [newname] Gallery
operation when using template database, default select
Help              Outputs this help message
Import            [filename] [all|setname] Imports template set from
channeltemplates.xml file into database
Info              [set] [type] [template] Outputs information of a templateset or a
single template
List              [set] [type] [regex] List templates / template sets
Open              {filename} Opens the given channeltemplates file
Quit              Terminates Mosart Template Set Editor
Rebase            [set] [baseset] [-c] [-f] Bases a templateset upon another set
Remove            [set] [type] [template]  Removes templateset or a single template
Rename            [set] [newname] Renames templateset
```

```
Save             Saves channeltemplates
SaveAs           [filename] [ChangesOnly] Saves channeltemplates to the given
filename. If ChangesOnly is given only changed templates are saved.
SetPrivate       [template set name] makes a shared template set private
SetShared        [template set name] makes a private template set shared
Verbose          Turns verbose logging to console on/off
```

Compare [set1] [set2] [type] [template]

Compares two template sets or a single template between two template sets. If any differences are encountered an external diff tool is launched for investigating the differences. Configuration of diff tool is done via the application configuration file, *TemplateSetEditor.exe.config*.
Syntax:

- **set1:** Name of template set one.
- **set2:** Name of template set two.
- **type:** Template type of single template (optional).
- **template:** Name or variant of single template (optional).

Example: Compare

```
# Compares template sets LDN and LDN2
compare LDN LDN2
# Compares template Camera+1 from template sets LDN and LDN2
compare LDN LDN2 0 1
compare LDN LDN2 Camera 1
```

Copy [set] [newset]

Makes a copy of a template set with name equal to newset.

Equals [set1] [set2] [type] [template]

Performs the equality test on two template sets or for a single template.
Syntax:

- **set1:** Name of template set one.
- **set2:** Name of template set two.
- **type:** Template type of single template (optional).
- **template:** Name or variant of single template (optional).

Example: Equals

```
# Performs the equality test on the template sets LDN and LDN2
equals LDN LDN2
# Performs the equality test on template Camera+1 from template sets LDN and LDN2
equals LDN LDN2 0 1
equals LDN LDN2 Camera 1
```

Edit [set] [type] [template]

Opens a single template in an external xml editor. The tool to use is configured in the application configuration file, TemplateSetEditor.exe.config
Syntax:

- **set:** Name of template set.
- **type:** Template type of single template.
- **template:** Name or variant of single template.

Example: Edit

```
# Opens template Camera+1 in template sets LDN in an external xml editor
edit LDN 0 1
edit LDN Camera 1
```

Flatten [set] [type] [template]

Flattens an entire templateset to its base set or a single template.
Syntax:

- **set:** Name of template set to flatten.
- **type:** Optional: Template type of single template (optional).
- **template:** Name or variant of single template (optional).

Example: Flatten

```
# Flattens template set LDN
flatten LDN
# Flattens template Camera+1 in template sets LDN
flatten LDN 0 1
flatten LDN Camera 1
```

Gallery [gallery|add|select|delete|rename|list] [gallery] [newname]

Gallery operations. Requires template database.

> **Tip:** This is the recommended way to remove a gallery from the template database. This operation is not possible with Template Editor.

Syntax:

- **operation:** Gallery operation. Either *add*, *select*, *delete*, *rename* or *list*. Default = *select*.
- **gallery:** Name of gallery in the Mosart template database.
- **newname:** New gallery name when operation=rename.

Example: Gallery

```
# List all galleries in the template database
gallery list
# Selects StudioC as the current gallery, both examples are valid
gallery StudioC
gallery select StudioC
# Add StudioR as new gallery to the template database
gallery add StudioR
# Deletes StudioC from the template database. This operation removes all content
related to the gallery.
gallery delete StudioC
# Renames StudioB to StudioB2
gallery rename StudioB StudioB2
```

Import [filename] [all | setname]

Imports template set from channeltemplates.xml file into database

Tip: The filename of the xml containing the sets/templates must not be called *channeltemplates.xml* if it resides in the c:\channeltemplates folder.
TemplateSetEditor overwrites this file when the app starts and the database is loaded. The file can be in any path and any name. (if not in c:\channeltemplates)

Syntax:

- **filename:** file name and path of channeltemplates.xml file to import
- **setname:** the individual set name to import or 'all' to import all sets

- All template set names imported will have a prefix of ***I_*** as to not duplicate if there is the same set name.
- Imported sets that are based on other set imported will also receive the prefix so all imported sets will match the existing hierarchy and not be separated.
- Directakes will be skipped (not imported)
- The console command **save** must be called to write back the imported sets/templates to the TDB after the **import** command.

Info [set] [type] [template]

Outputs the XML content of a template set or a single template to the console and the Windows clipboard.

Tip: This operation copies the XML content of a single template to the Windows clipboard.

Syntax:

- **set:** Name of template set.
- **type:** Template type of single template (optional).
- **template:** Name or variant of single template (optional).

Example: Info

```
# Outputs information of template set LDN
info LDN
# Outputs information of template Camera+1 in template sets LDN
info LDN 0 1
info LDN Camera 1
```

List [set] [type] [regex] [file=filename]

List all template sets or templates within a template set.
Syntax:

- **set:** Template set to list templates from (optional).
- **type:** Restrict listing to templates of given types (optional).
  - '*' = All types
  - Comma separated list of valid types.
- **regex:** Restrict listing to templates matching the given regular expression (optional).
- **filename:** File name for storing listed templates (optional).

Example: List

```
# List all template sets within the current channeltemplates.xml file
list
# List all templates within template set LDN
list LDN
# List all camera templates within template set LDN
list LDN 0
list LDN Camera
# List all camera templates within template set LDN where name contains "RIGHT"
list LDN 0 RIGHT
list LDN Camera RIGHT
# List all camera and live templates within template set LDN where name contains
"RIGHT"
list LDN 0,3 RIGHT
list LDN Camera,Live RIGHT
# List all templates within template set LDN where name contains "RIGHT"
list LDN * RIGHT
```

Open [filename]

Opens tempateset / *channeltemplates.xml* file. Default: *C:\channeltemplates\ChannelTemplates.xml.*

Rebase [set] [baseset] [-c] [-f]

Bases a template set upon another template set. If a base set is not given, the given template set is made *standalone* (is not based upon any other template set).
Syntax:

- **set:** The template set to rebase.
- **baseset:** The template set to used as the base set.

- **-c:** Makes a copy of the base set if the given templateset does not exist.
- **-f:** Flattens template set after the *rebase* operation.

Example: Rebase

```
# Bases template set LDN2 to template set LDN
rebase LDN2 LDN
# Bases template set LDN2 to template set LDN.
# If LDN2 does not exist, a copy of LDN is made before rebasing
rebase LDN2 LDN -c
# Bases template set LDN2 to template set LDN.
# After rebase, a flattening templateset operation is performed
rebase LDN2 LDN -f
# Makes template set LDN2 'stand alone' (i.e. not based upon any other template set).
rebase LDN2
```

Remove [set] [type] [template]

Removes the given template set or a single template.

Syntax:

- **set:** Name of template set
- **type:** Template type of single template (optional).
- **template:** Name or variant of single template (optional).

Example: Remove

```
# Removes template set LDN
remove LDN
# Removes template Camera+1 from template sets LDN
remove LDN 0 1
remove LDN Camera 1
```

Rename [set] [newname]

Renames the given template set to the given new name.

Save

Saves any changes to the template database and the current *channeltemplates.xml* file.

SaveAs [filename]

Saves the current templatesets to the given filename.

SetPrivate [template set name]

Set a current shared template set to private in the currently selected gallery

Syntax:

- **template set name:** the name of the template set to make private

SetShared [template set name]

Set a current private template set in the currently selected gallery to shared

Syntax:

- **template set name:** the name of the template set to make shared

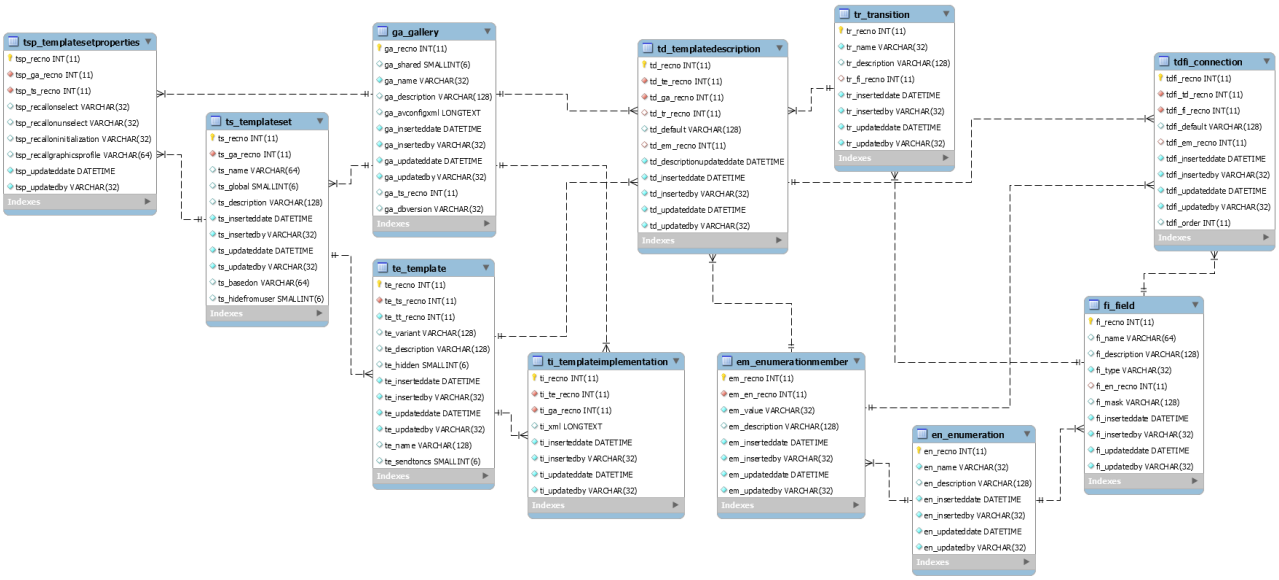
13.8 Viz Mosart Template Database Specification

This section provides database specifications for the Viz Mosart Template Database:

- Introductions and Notations
- Viz Mosart Database Tables

13.8.1 Entity Relationships

The following figure shows the relationships between the most important tables within the database.



13.8.2 Introductions and Notations

By default, Viz Mosart organizes all templates in a single file, *C:\channeltemplates\ChannelTemplates.xml*.

The Template Database enables storing of templates in a MySQL database.
The main benefits are the ability to:

- Share templates between Mosart installations and galleries.
- Provide synchronized template sets between the Mosart main and backup servers.

Viz Mosart Template database tables

Table name	Prefix	Short description
db_database	db	Mosart Template Database update table. Contains an overview of database update.
em_enumerationmember	em	The enumeration members that may be used as values for template fields of enumeration type.
en_enumeration	en	The enumerations that may be used as types (i.e. sets of admissible values) for template fields.
fi_field	fi	The fields used as template inputs.
ga_gallery	ga	The galleries.
lo_lock	lo	Contains locks for editing database content.
sq_sequence	sq	Contains sequences stored as part of the Mosart template sets.
ta_templatetypealias	ta	The template type aliases (a template type may be referred to by several aliases).
td_templatedescription	td	The template descriptions. Input fields and transition input fields. A template may have different descriptions for different galleries.
tdfi_connection	tdfi	Relation, relating template description and fields.

Copyright © 2026 Vizrt

Table name	Prefix	Short description
te_template	te	The templates.
ti_templateimplementation	ti	The template implementations. A template may have different implementations for different galleries.
tr_transition	tr	Available transitions to be applied between templates.
ts_templateset	ts	The template sets (the templates are organized in sets).
tsp_templatesetproperties	tsp	Template set properties
tt_templatetype	tt	The Mosart template types.

Conventions and Notations

Table Prefix

Each table has a unique prefix made by abbreviating the rest of the table name, followed by an underscore (_). This prefix is used both in the table name and for all columns in the table.

Using a prefix helps in identifying the relationships between column names and tables.

Syntax:

Table	<prefix>_<tableName>	ts_templateset
Column	<prefix>_<columnName>	ts_recno, ts_name
Foreign key	<prefix>_<foreignColumnName>	ts_ga_recno

Primary Keys

Each table has an integer column *<prefix>_recno* representing a Primary key, and thereby as value of foreign keys, where applicable.

Foreign Keys

A foreign key (referencing a *<prefix>_recno* column of some table) usually has the name …_…_recno (with the two …s representing the prefixes, still except the _, of the containing and referenced tables, respectively).

Info: Where a table has more than one Foreign key to the same table, you will need to create an alternative name.

Data Types

Type	Comment
datetime	Data type capable of storing a timestamp (date and time of day).
varchar(n)	Variable length string.
integer	General integer.
smallint	Small integer, normally used to store boolean values (`0` =false, `1` =true).
longtext	Data type capable of storing a Unicode string of virtually any length.

Fixed Columns

Every table contains the following four fixed columns to store update and insert information for the individual rows. Note that these columns are not stated in the accompanying table descriptions.

Name	Type	Allow Null	Description
<prefix>_inserteddate	datetime	No	When the row was inserted.
<prefix>_insertedby	varchar(32)	No	Mosart server that handled the insert. Provided by AV Automation property *TemplateDbDefaultInserter*.
<prefix>_updateddate	datetime	No	When the row was last updated/inserted.

13.8.3 Viz Mosart Database Tables

This section contains a brief overview of all tables contained in the Viz Mosart Database schema.

- db_database
- em_enumerationmember
- en_enumeration
- fi_field
- ga_gallery
- lo_lock
- sq_sequence
- ta_templatetypealias
- td_templatedescription
- te_template
- ti_templateimplementation
- ts_templateset
- tt_templatetype
- tdfi_connection
- tr_transition
- tsp_templatesetproperties
- vtefi_connectionX

Copyright © 2026 Vizrt

db_database

The *db_database* table is a maintenance table that contains the current version of the Viz Mosart Template Database along with all its upgrades.

The table is used by Mosart AV Automation to verify whether an automatic upgrade is required.

Name	Type	Allow Null	Description
db_recno	integer	No	Primary key.
db_version	integer	No	The database version number.
db_versiondate	varchar(32)	No	The date when the upgrade was introduced to the Mosart Template Database.
db_mosartversion	varchar(32)	Yes	The Viz Mosart version number responsible for doing the upgrade.

Copyright © 2026 Vizrt

em_enumerationmember

The em_enumerationmember contains the enumeration values for the corresponding enumeration types.

Name	Type	Allow Null	Description
em_recno	integer	No	Primary key.
em_en_recno	integer	No	Foreign key referencing the corresponding enumeration in the *en_enumeration* table.
em_value	varchar(32)	No	The value of the enumeration member. Normally presented to the users.
em_description	varchar(128)	Yes	Optional enumeration member description.

Copyright © 2026 Vizrt

en_enumeration

The table *en_enumeration* contains all enumerations used to set values for template fields.

Name	Type	Allow Null	Description
en_recno	integer	No	Primary key.
en_name	varchar(32)	No	The enumeration name. Normally used in drop-down lists presented to users.
en_description	varchar(128)	Yes	Optional enumeration description.

fi_field

The table *fi_field* contains valid fields or newsroomtags used by Mosart templates for receiving user input. Normally these fields are exposed to the uses in the NRCS via dedicated ActiveX components. Note that a single field can be shared by many templates.

Name	Type	Allow Null	Description
fi_recno	integer	No	Primary key.
fi_name	varchar(32)	No	The field name. Normally presented to users.
fi_description	varchar(128)	Yes	Optional field description.
fi_type	varchar(32)	No	The field type. Supported values are ‘ENUMERATION’, ‘NUMBER’, ‘STRING’.
fi_en_recno	integer	Yes	Optional foreign key for ENUMERATION fields. If set, referencing the corresponding enumeration in the en_enumeration table.
fi_mask	varchar(128)	Yes	Optional input mask. Used for STRING fields only.

Copyright © 2026 Vizrt

ga_gallery

The table *ga_gallery* contains all Mosart galleries using the database.

> **Note:** In Mosart AvAutomation the property *ThisGallery* shall reflect the gallery name as stored in the *ga_name* column.

Name	Type	AllowNull	Description
ga_recno	integer	No	Primary key.
ga_shared	smallint	Yes	Set to true (1) for the *shared* gallery. The *shared* gallery may be used by clients to obtain all shared template sets within the database. Only one gallery shall be used as the *shared* gallery. By default, a virtual gallery named SHARED is used for this purpose.
ga_name	varchar(32)	No	The gallery name.
ga_description	varchar(128)	Yes	Optional gallery description.
ga_avconfigxml	longtext	Yes	The XML formatted contents of the *avconfig.xml* file used in this gallery.
ga_ts_recno	integer	Yes	Optional foreign key referencing the ‘default’ template set of the gallery.

Name	Type	AllowNull	Description
ga_dbversion	string	Yes	The database version used by the Mosart applications for the gallery. This database version reflects the current Mosart version.

Note: One row only should have *ga_shared* = 1. The name of this gallery is by default set to 'SHARED'. All shared templates belong to this gallery.

lo_lock

The table *lo_lock* contains information used to restrict editing access to a template. The template database locks at template level (i.e. only one user may edit a single template at any time).

The lock table also contains rows used to store timestamps for last updates within a single gallery or the database overall. These values may be used to poll the database for changes. These timestamps are stored in the *lo_updatedate* field.

Note that the *lo_insertedby* field is used to identify the lock type:

- **Template:** This is a true lock, locking the corresponding template. The template is identified by the value in the *lo_recno* column which contains a foreign key to the *te_recno* value in the *te_template* table.
- **Gallery:** Contains a timestamp when the corresponding gallery was last updated. The gallery is identified by the *lo_recno* which for gallery types contains a foreign key to the corresponding gallery, *ga_recno* in the *ga_gallery* table. The foreign key is a function of the *lo_recno* value as follows: ga_recno = 10 - lo_recno
- **Database:** Contains a timestamp when the database was last updated.
- Other values are deprecated.

Name	Type	Allow Null	Description
lo_recno	integer	No	Primary key. Also used to identify either a gallery or a template depending upon the lock type. See description above.
lo_ga_recno	integer	Yes	Optional foreign key referencing the gallery owning the lock.
lo_ga_name	varchar(32)	Yes	The name of the gallery owning the lock.
lo_insertedby	varchar(32)	No	Contains the lock type. Either Template, Gallery or Database. See description above.

sq_sequence

The *sq_sequence* table contains recorded sequences from a Viz Mosart Automation. Sequences are gallery dependent and related to the template set in use when recorded. A sequence makes it possible to record any sequence from a rundown in Viz Mosart automation for later playback.

Name	Type	Allow Null	Description
sq_recno	integer	No	Primary key.
sq_ga_recno	integer	No	Foreign key referencing the corresponding gallery *ga_recno* in the *ga_gallery* table.
sq_ts_recno	integer	No	Foreign key referencing the corresponding template set *ts_recno* in the *ts_templateset* table.
sq_name	varchar(32)	No	Name of sequence.
sq_description	varchar(128)	Yes	Optional description of the sequence.
sq_xml	longtext	No	The gallery specific implementation of the sequence. Stored as received from Mosart Automation in XML.

ta_templatetypealias

The *ta_templatetypealias* table contains custom user aliases for the Mosart template types. These aliases are used by Mosart ManusAdministrator to translate user aliases to the corresponding Mosart template types.

Name	Type	AllowNull	Description
ta_recno	integer	No	Primary key.
ta_name	varchar(32)	No	The template type alias name. Should be unique, as this is presented to the users.
ta_description	varchar(128)	Yes	Optional template type alias description.
ta_tt_recno	integer	No	Foreign key referencing the corresponding Mosart template type in the *tt_templatetype* table.
ta_default	smallint	Yes	A value of 1 is used for the 'default' alias among those belonging to the same template type. For each template type, exactly one alias belonging to that type should have ta_default = `1`.

Information: The ta_templatetypealias table is the only table that is maintained and used by Mosart ManusAdministrator containing the NRCS template type aliases.

td_templatedescription

The table *td_templatedescription* contains template information to be presented to users that may differ among the galleries (i.e. this information is gallery dependent).

Name	Type	Allow Null	Description
td_recno	integer	No	Primary key.
td_te_recno	integer	No	Foreign key referencing the corresponding template *te_recno* in the *te_template* table.
td_ga_recno	integer	No	Foreign key referencing the corresponding gallery *ga_recno* in the *ga_gallery* table
td_tr_recno	integer	Yes	Optional foreign key referencing a transition *tr_recno* in the *tr_transition* table.
td_default	varchar(128)	Yes	Optional default value for transitions not of type ENUMERATION. Normally for MIX and WIPE transitions.
td_em_recno	integer	Yes	Optional foreign key for the default value for ENUMERATION transition types. Normally EFFECT transition. Reference to a enumeration value em_recno in the em_enumerationmember table.
td_descriptionupdateddate	Datetime	No	(deprecated)

Note: The overall template description is stored in three parts / tables:

1. **td_templatedescription:** The base template description.
2. **tr_transition:** The transition used by the template. In Mosart this equals the transition when taking the template. Normally either Mix, Wipe or Effect.
3. **tdfi_connection:** Relation that list all fields (or newsroomtags) used by the templates. A newsroomtag is a template property allowing users to input information to the template. Normally newsroomtags are exposed to the users in the NCS.

Copyright © 2026 Vizrt

te_template

The table *te_template* contains the individual templates with information that is shared among all galleries.

Note that information regarding a single template is divided into three parts / tables:

1. **te_template:** Contains basic template information. This information is shared among all galleries (i.e. gallery independent).
2. **td_templatedescription:** Contains typically template information to be presented to users that is gallery dependent.
3. **ti_templateimplementation:** Contains the template implementation as created by the Mosart template editor. This information is gallery dependent.

Name	Type	Allow Null	Description
te_recno	integer	No	Primary key.
te_ts_recno	integer	No	Foreign key referencing the corresponding template set in the *ts_templateset* table.
te_tt_recno	integer	No	Foreign key referencing the corresponding template type in the *tt_templatetype* table.
te_variant	varchar(32)	No	The Mosart template variant.
te_description	varchar(128)	Yes	Optional template description.
te_hidden	smallint	Yes	Set to true (1) if this template is to be hidden from any user presentations.
te_name	varchar(128)	Yes	Optional template name.
te_sendtoncs	smallint	Yes	Set to true (1) if the template implementation is to be sent to NCS. Used by some NCS only.

ti_templateimplementation

The table *ti_templateimplementation* contains the gallery specific template implementation as created by the Mosart template editor (i.e. this information is gallery dependent).

Name	Type	Allow Null	Description
ti_recno	integer	No	Primary key.
ti_te_recno	integer	No	Foreign key referencing the corresponding template *te_recno* in the *te_template* table.
ti_ga_recno	integer	No	Foreign key referencing the corresponding gallery *ga_recno* in the *ga_gallery* table.
ti_xml	longtext	Yes	The actual xml-formatted template implementation as created by the Mosart template editor.

Information: The template implementation must be created per gallery and is stored using its native XML representation, as received from the Mosart template editor

ts_templateset

The table *ts_templateset* contains the template sets.

Name	Type	Allow Null	Description
ts_recno	integer	No	Primary key.
ts_ga_recno	integer	No	Foreign key referencing the corresponding gallery in the *ga_gallery* table. Template sets belonging to the shared gallery (*ga_shared*=1) are by definition *shared* template sets. Shared template sets will be present in all galleries. Template sets belonging to any other gallery are by definition *private* template sets owned by the corresponding gallery.
ts_name	varchar(64)	No	The name of the template set.
ts_global	smallint	Yes	(deprecated)
ts_description	varchar(128)	Yes	Optional template set description.
ts_ga_updating	integer	Yes	(deprecated)
ts_basedon	varchar(64)	Yes	Hierarchical template sets. Specifies the base template set, if applicable.
ts_hidefromuser	smallint	Yes	Non-zero (true) if the template set shall be hidden from users (i.e. from Mosart GUI and NCS presentations).

tt_templatetype

The table *tt_templatetype* contains the Mosart template types.

Name	Type	Allow Null	Description
tt_recno	integer	No	Primary key.
tt_name	varchar(32)	No	The template type name. Should be unique, as this is presented to the users.
tt_description	varchar(128)	Yes	Optional template type description.

The Mosart template types

Id: tt_recno	Name: tt_name	Description
-1	UNKNOWN	
0	CAMERA	
1	PACKAGE	
2	VOICEOVER	
3	LIVE	
4	FULLSCREENGRAPHICS	
5	DVE	
6	JINGLE	
7	TELEPHONEINTERVIEW	
8	ADLIBPIX	
9	BREAK	
10	MACRO	
17	VIDEOWALL	

Copyright © 2026 Vizrt

Id: tt_recno	Name: tt_name	Description
25	CLIP	
100	LOWERTHIRDS	
110	OVERSHOULDERGRAPHICS	
215	PLAYSOUND	
216	SETSOUNDLEVEL	
217	LIGHT	
220	ACCESSORIES	
250	SETCROSSPOINT	
275	COMMAND	
310	EXTERNALCOMMAND	
320	TEXTTELEPROMPTER	
330	CLIPDUR	
331	ENDFRASE	
332	STORYNR	
333	ATTACHA	
334	CLIPSTART	
335	CLIPHIRESPATH	
340	GRAPHICSID	
350	MARKER	
360	TIMINGINFO	
999	IGNORE	

Copyright © 2026 Vizrt

tdfi_connection

The table *tdfi_connection* which contains a list of fields (or newsroomtags) that corresponds to a template description. See also information given in the td_templatedescription table.

This is a relation table defining a many-to-many relationship between the *td_templatedescription* and *fi_field* tables.

Name	Type	Allow Null	Description
tdfi_recno	integer	No	Primary key.
tdfi_td_recno	integer	No	Foreign key referencing the corresponding template description td_recno in the td_templatedescription table.
tdfi_fi_recno	integer	No	Foreign key referencing the corresponding field fi_recno in the fi_field table.
tdfi_default	varchar(128)	Yes	Optional default value for fields not of type ENUMERATION. I.e for STRING and NUMBER types.
tdfi_em_recno	integer	Yes	Optional foreign key for the default value for ENUMERATION field types. Reference to a enumeration value em_recno in the em_enumerationmember table.

Copyright © 2026 Vizrt

tr_transition

The table *tr_transition* contains the transition types available for Mosart templates. A transition is defined as the transition to take when taking a template (the transition from the former template).

Name	Type	Allow Null	Description
tr_recno	integer	No	Primary key.
tr_name	varchar(32)	No	The transition name.
tr_description	varchar(128)	Yes	Optional transition description.
tr_fi_recno	integer	Yes	Optional foreign key referencing an input field, *fi_recno* in the *fi_field* table.

Copyright © 2026 Vizrt

tsp_templatesetproperties

The *tsp_templatesetproperties* table contains some optional gallery specific template set properties.

Name	Type	Allow Null	Description
tsp_recno	integer	No	Primary key.
tsp_ga_recno	integer	No	Foreign key referencing the corresponding gallery ga_recno in the ga_gallery table.
tsp_ts_recno	integer	No	Foreign key referencing the corresponding template set ts_recno in the ts_templateset table.
tsp_recallonselect	varchar(32)	Yes	Optional direct take to be recalled when the template is selected in Viz Mosart automation.
tsp_recallonunselect	varchar(32)	Yes	Optional direct take to be recalled when the template is unselected in Viz Mosart automation.
tsp_recalloninitialization	varchar(32)	Yes	Optional direct take to be recalled when a rundown is reloaded.
tsp_recallgraphicsprofile	varchar(32)	Yes	Optional direct take to be recalled when a graphics profile (or graphics concept) is selected in Viz Mosart automation.

Copyright © 2026 Vizrt

13.9 Configuring the Viz Mosart Server

Both AV Automation and Manus Administrator can be configured to use the Viz Mosart Template Database.

- **AV Automation**: For storing templates and the xml configuration file, AvConfig.
- **Manus Administrator**: For storing NRCS *template type aliases* as configured in the newsroomsettings NCS tags editor.

> ⚠ No other information, other than NRCS template type aliases from *newsroomsettings.xml* is stored in the database.

Both applications extract information from the database and store the information locally on file:

- **AV Automation**: In files *channeltemplates.xml* and *avconfig.xml*.
- **Manus Administrator**: In file *newsroomsettings.xml*.

> ⚠ Any existing files will be overwritten. Before connecting to the database for the first time, always back-up these three files.

The following topics are presented:

- AV Automation and the Mosart Template Database
  - Connection Setup
  - AV Automation Actions
  - Importing channeltemplates.xml
  - Importing avconfig.xml
- Manus Administrator and the Mosart Template Database
  - Connection Setup
  - Import newsroomsettings.xml
- Mosart Web Apps NRCS Plugin
- Viz Mosart ActiveX

13.9.1 AV Automation and the Mosart Template Database

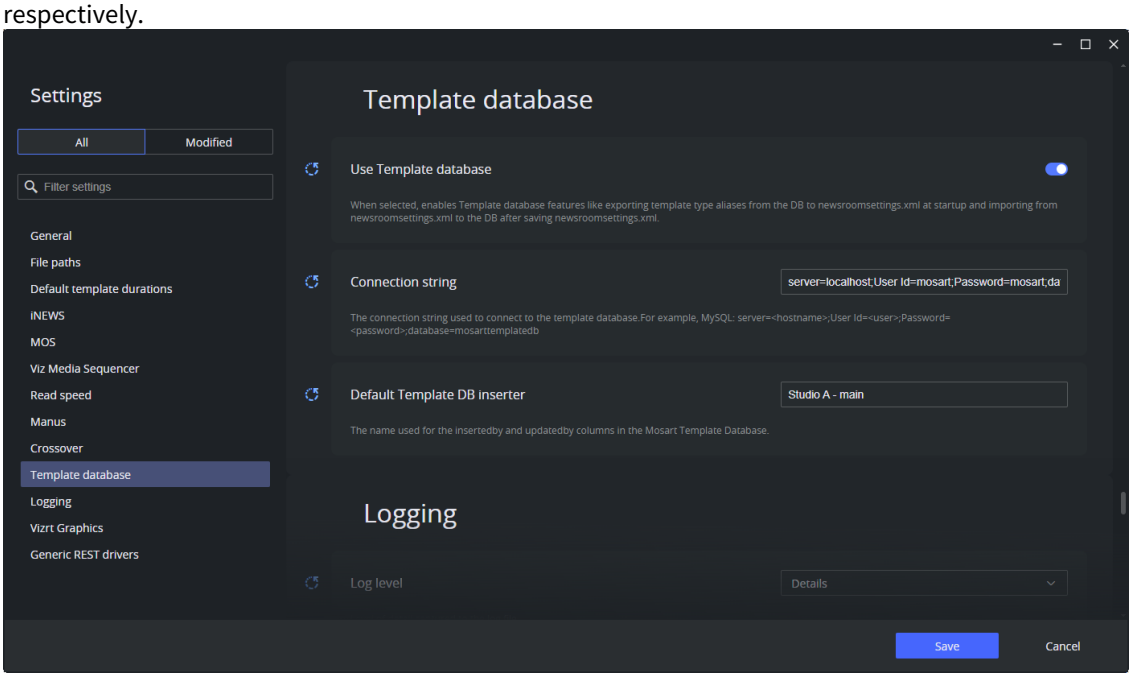
Connection Setup

To connect AV Automation to the Template Database

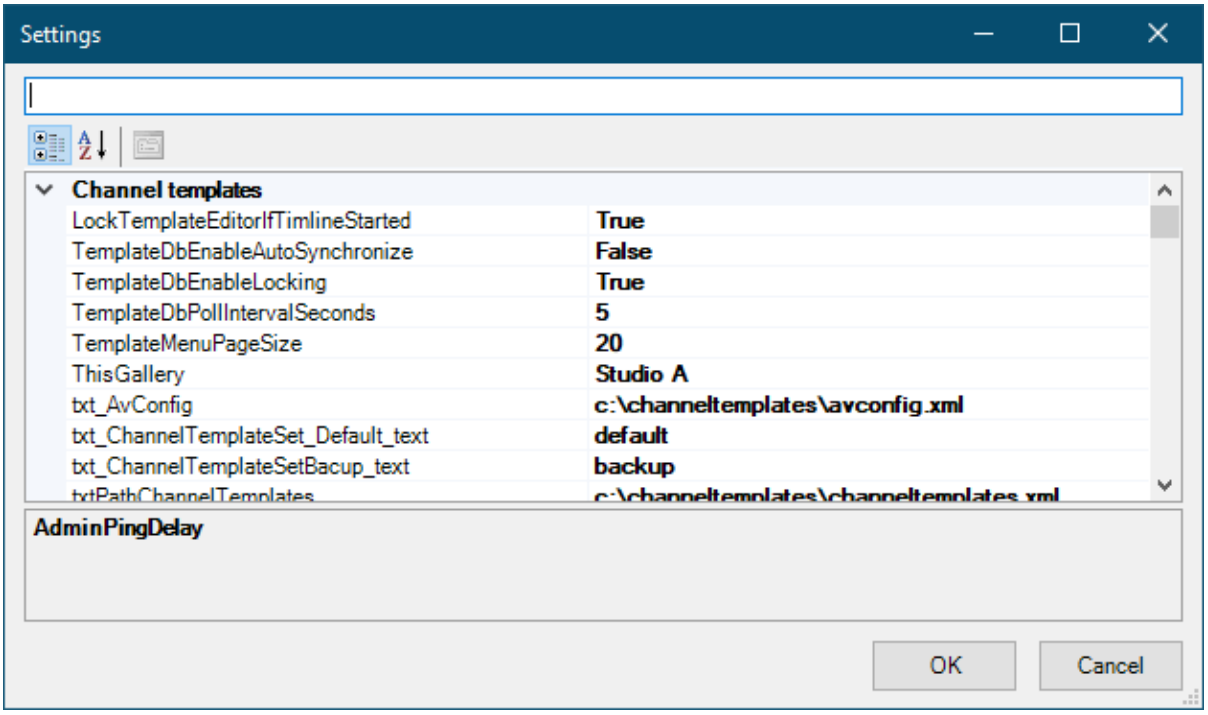
1. With AV Automation running, navigate from the menu bar to the Settings dialog, **Devices > Settings** (or keyboard shortcut combination **CTRL + SHIFT + P**).
2. Ensure these settings are made:
   a. **Use Template database**: Set to *True* to enable connection to the Viz Mosart Template Database (Default: *False*).
   b. **Connection string**: The connection string used to connect to the template database.
   For example, for a MySQL database:

```
server=<hostname>;User
Id=<user>;Password=<password>;database=mosarttemplatedb
```

c. **Default Template Db Inserter**: The name to be used for the *…_insertedby* and *…_updatedby* columns, the entities enabling a database update.
These names are only visible in the database itself where the *_insertedby* and *_updatedby* columns are populated with this property.
We recommend to specify as *ThisGallery* + main/backup, to identify main and backup servers respectively.



3. Click **Save**.
4. The following additional settings are set using the old settings dialog (accessed with **CTRL + SHIFT + S**):



a. **ThisGallery**: The name of the DB gallery representing the gallery of this Viz Mosart installation. This DB gallery need not exist, but will be inserted when saving in A/V Setup or saving to DB in the Template Editor.
The value must be identical for both the Mosart main and backup servers, with a unique name for each gallery (Default: *NONSHARED*).

b. **SharedGallery**: This setting SHOULD NOT be changed from its default value, and not be changed using the Settings dialogue.
However, it *can* be changed in the *MMAVAutomation.exe.config* file, but this is *not* recommended (Default: *SHARED*).
SharedGallery is the DB name of the pseudo-gallery containing the shared template sets.
If Viz Mosart has been installed to use the Template DB functionality, this gallery has already been inserted into the DB.

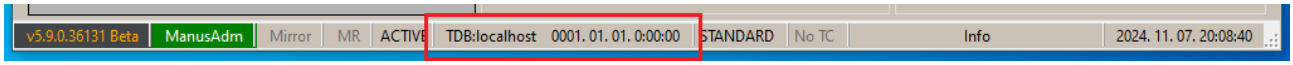
c. **TemplateDbEnableAutoSynchronize**: When *true,* idle Mosart Servers targeting the same gallery (*ThisGallery*) are automatically synchronized by AV Automation.
Changes made to a template on one Mosart server are automatically obtainable by the other Mosart servers.
This is typically used by Mosart main/backup servers. Recommended value: *True* (unless your show design has high-intensity template changes. See TemplateDbPollIntervalSeconds).
Note: Only *idle* servers will update automatically. *Live* servers requires a manual operation, for example by opening the Template Editor, clicking on the **Status** bar or by restarting AV Automation.

d. **TemplateDbPollIntervalSeconds**: Number of seconds between Mosart server polling the database for updates.
Used by the synchronization mechanism to check whether a synchronization operation is necessary.
If set to *0*, 5 seconds is used. Recommended value: 5 seconds (Default: *0*, giving 5 seconds).
Set to a higher value than the default (5 seconds) if multiple frequent template changes occur.

e. **TemplateDbEnableLocking** (deprecated): Leave at default value (*True*). Only used on legacy systems, to ensure only one Mosart server gets write access to a template in the template database at a time, preventing two users from simultaneously editing the same template.

5. Restart AV Automation.

When successful, an entry for the gallery is added to the database and initial templates (*channeltemplates.xml*) and AV Config (*avconfig.xml*) are uploaded.
Successful connections is also indicated in the AV Automation status bar:



To verify AV Automation database connection

1. Perform connection steps above, To connect AV Automation to the Template Database.
2. Using any MySql database client, verify the content of the *ga_gallery* table. This table shall now contain a single row where *ga_name*=ThisGallery.

AV Automation Actions

When connected to the Template Database, AV Automation performs state-dependent actions.

At AV Automation launch

- avconfig.xml is replaced by the value tied to *ThisGallery*.
  - If *ThisGallery* does not yet exist, any existing avconfig.xml is used as is, enabling an existing setup for template sharing.
- channeltemplates.xml is replaced.
  - Any shared template sets are written as channels elements to the new channeltemplates.xml with a gallery attribute of *SharedGallery*
  - Any template sets tied to *ThisGallery* are written as channels elements to the new channeltemplates.xml with a gallery attribute of ThisGallery
  - If *ThisGallery* does not yet exist, any channels elements in the old channeltemplates.xml (except those with a gallery attribute of *SharedGallery*), are transferred as-is to the new one.
    This avoids naming conflicts and facilitates bringing an existing setup to template sharing.

Saving to A/V Setup

- If *ThisGallery* does not exist in the database, it is inserted.
- The new contents of avconfig.xml are stored in the database tied to *ThisGallery*.
- The contents of selected enumerations are stored separately in the database.
  - These are used when storing channeltemplates.xml in the database
  - A new or changed A/V Setup must always be saved, before saving to the template database.

Saving to Template Database

- If *ThisGallery* does not exist in the database, it is inserted.
- The new content of channeltemplates.xml is stored in the database.
  Channel elements, Template Sets, with a gallery attribute of *SharedGallery* are stored as shared Template Sets - for example, tied to *SharedGallery*.
  Each template in these sets has:

  - A template description tied to *SharedGallery*
  - A template description tied to *ThisGallery*, the two descriptions will be equal
  - A template implementation tied to *ThisGallery*.

Any existing descriptions and implementations tied to other galleries are retained.

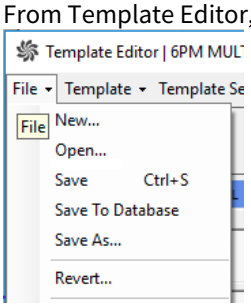
- Other channels elements, including any new template sets are tied to *ThisGallery*. Each template in these sets will have:
  - A template description tied to *ThisGallery*
  - A template implementation tied to *ThisGallery*.

Importing channeltemplates.xml

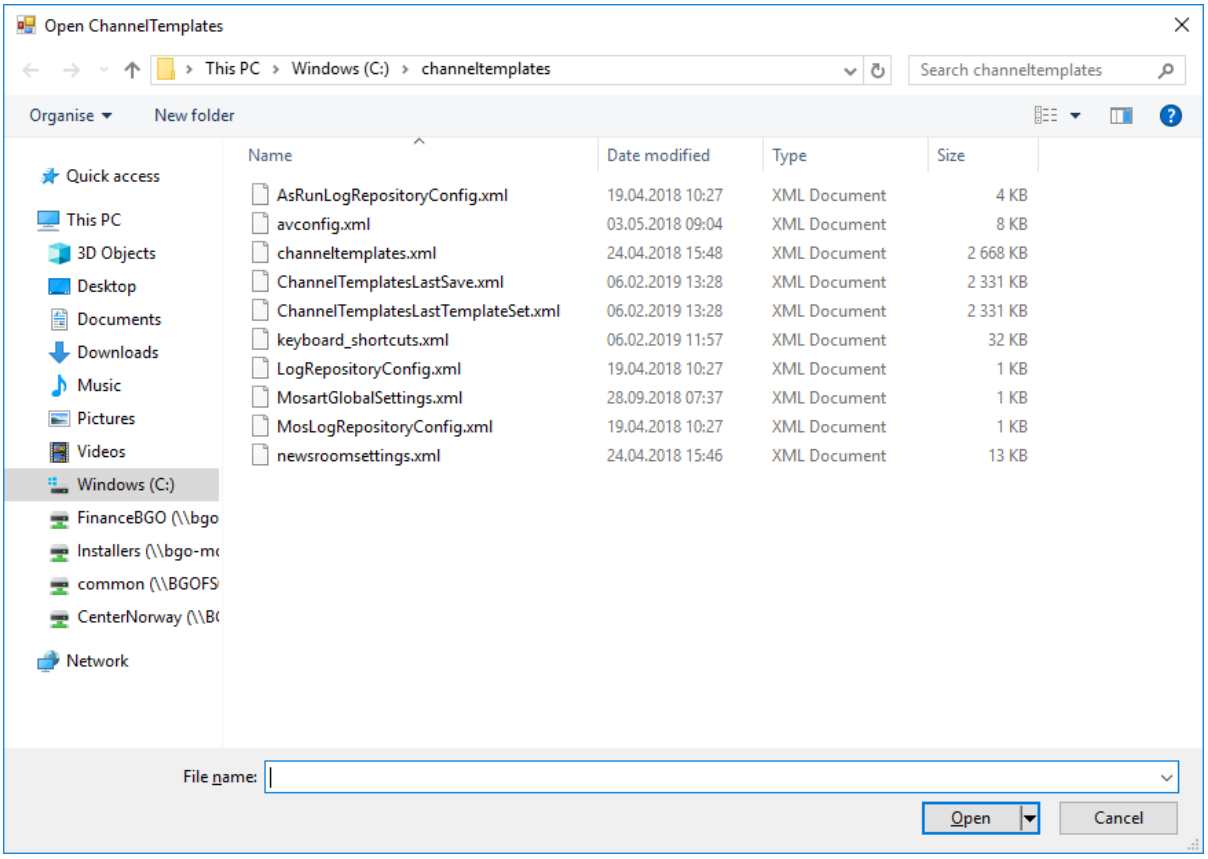
You can use a ready-made template set, stored as an XML-file. and manually import to the database. This is useful for populating the template database with content from a *channeltemplates.xml* file.
Either initially or as backup / transfer operation.

To import channeltemplates.xml to the Template Database

1. With AV Automation running, navigate from the menu bar to **Devices > Template editor**.
2. From Template Editor, open any *channeltemplates.xml* from file with **File >Open**.



3. Select required XML file from the list.



4. Click **Open**.

Note: After opening a template file in this way, **File > Save** will *not save to the database*, but to the **opened file**.
Only after completing next step **File > Save To Database** below, does this operation save to the template database.

5. Select the **File > Save To Database**.
   The content of the selected *channeltemplates.xml* file is now uploaded to the Template Database.

Note: When importing a *channeltemplates.xml* to the database any existing templates are overwritten. No templates are deleted from the database.

Importing avconfig.xml

You can manually import an *avconfig.xml* file to the database. *avconfig.xml* is stored as-is related to the gallery. Additionally, the lists of vision mixer effects and crosspoints, audio mixer inputs, video router sources and destinations are extracted and stored as global lists within the database. This information is used by the Viz Mosart ActiveX and Viz Mosart NRCS Plugin.

To import avconfig.xml to the Template Database

1. With AV Automation running, navigate from the menu bar to **Devices > A/V Setup**.
2. From the A/V Setup menu bar, select **File > Open** to open required *avconfig.xml* file.

3. With the required file visible in A/V Setup, select the **File > Save**.
   The content of the selected *avconfig.xml* file is now stored in the template database.

> **Note:** The content of the *avconfig.xml* file is stored as-is, a character string, in the *ga_gallery* table.

13.9.2 Manus Administrator and the Mosart Template Database

The only information stored in the database from Manus Administrator are the newsroom aliases for Mosart template types, '*NCS tags*' in the newsroomsettings editor.

This is done to make the newsroom aliases the same for all galleries connected to the database. This information is global and shared among all galleries. Therefore there is no gallery-specific setting necessary within Manus Administrator (unlike *ThisGallery* for AV Automation)

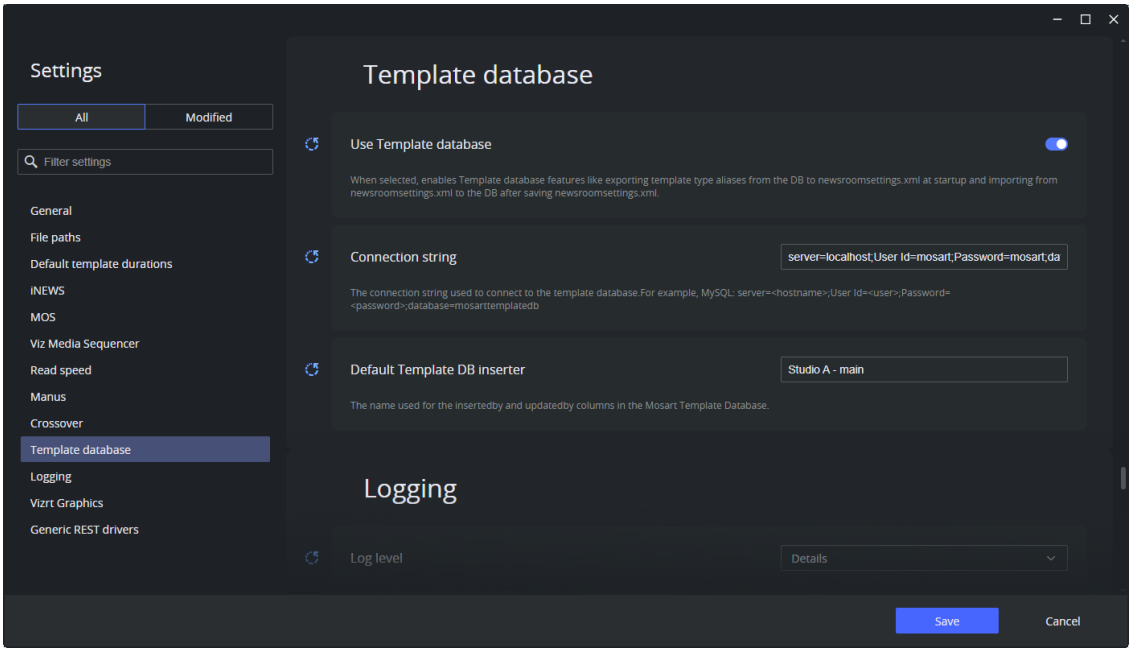
Connection Setup

To connect Manus Administrator to the Template Database

1. With Manus Administrator running, type `settings` followed by **ENTER**.
2. Ensure these settings are made:
   a. **Use Template database**: Set to *true* to enable connection to the Viz Mosart Template Database (Default: *false*).
   b. **Connection string**: The connection string used to connect to the template database.
   For example, for a MySQL database:

   ```
   server=<hostname>;User
   Id=<user>;Password=<password>;database=mosarttemplatedb
   ```

   c. **Default Template Db Inserter**: The name to be used for the *…_insertedby* and *…_updatedby* columns, the entities enabling a database update.
   These names are only visible in the database itself where the *_insertedby* and *_updatedby* columns are populated with this property.
   We recommend to specify as *ThisGallery* + main/backup, to identify main and backup servers respectively.



3. Click **Save**.

Import newsroomsettings.xml

To import newsroomsettings.xml to the Template Database

You can import NRCS template type aliases that are stored in *newsroomsettings.xml,* to the database.

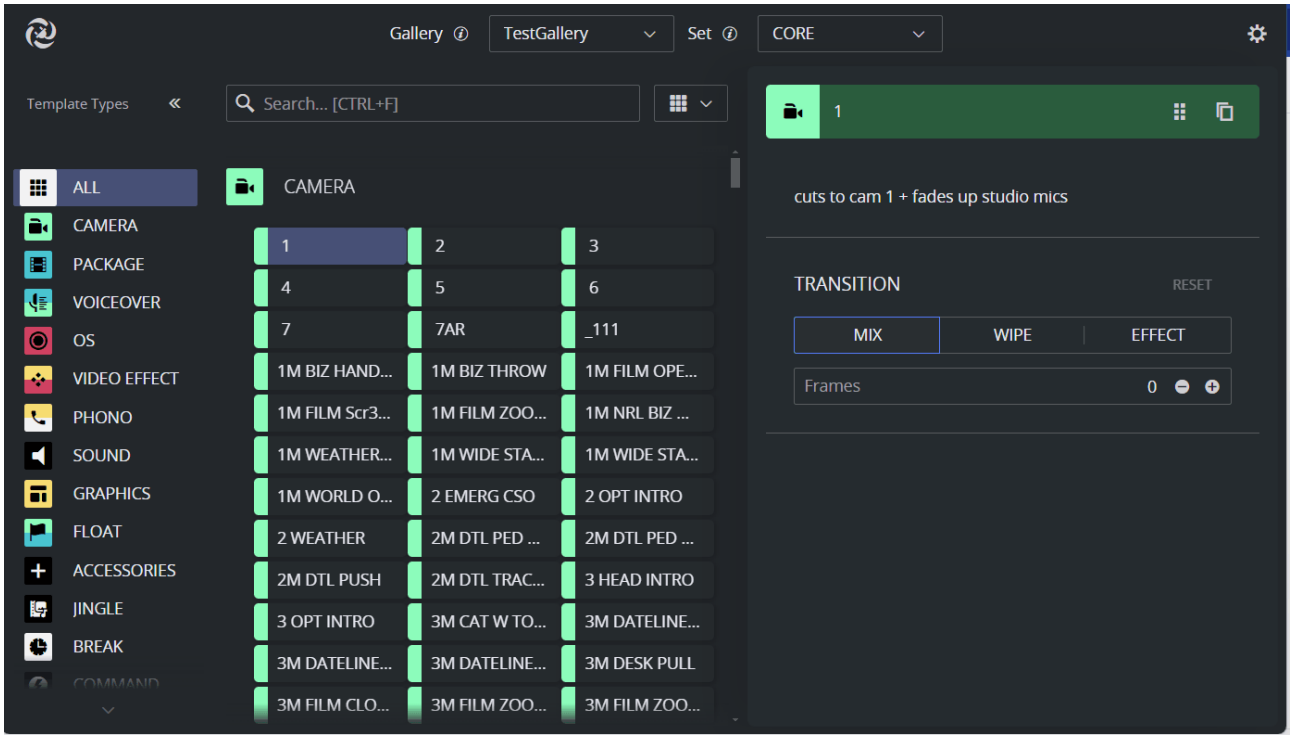
1. With Manus Administrator running, type *newsroomsettings*.
   This opens a separate **Newsroom Settings** menu.
2. From the menu bar, select **File > Open**.
3. Select required newsroomsettings.xml file and click **Open**.
4. Select the **File > Save**.

The newsroom aliases for template types are now uploaded to the Template Database.

> **Note:** It is only the *template type aliases* that are stored in the template database. All other information remains locally as is the *newsroomsettings.xml*.

13.9.3 Mosart Web Apps NRCS Plugin

Journalists use the Mosart Web Apps NRCS Plugin to insert and customize Viz Mosart template information into their NRCS script.



- From within their NRCS, the journalist searches for a Viz Mosart template, customizes the content, then drag and drops it into their script.
- When the script is saved, details are pushed to Manus Administrator, creating a corresponding entry in the active rundown in the Viz Mosart GUI.

> ⚠ **Note**: The Mosart Web Apps NRCS Plugin must be connected to your Viz Mosart server.

Connection is handled in a simple configuration wizard at Web App installation, offering one of two connection choices:

1. The Template Database.
2. Directly to the Viz Mosart Server.

For more information, see the latest Mosart Web Applications Guide.

13.9.4 Viz Mosart ActiveX

The Viz Mosart ActiveX is a legacy method for inserting Viz Mosart template information into the NRCS script. This product is superseded by the Mosart Web Apps NRCS Plugin.

- The user selects a Viz Mosart type and variant to create an entry in the Viz Mosart rundown.
- When the script is saved, information is pushed to Manus Administrator and onto the active rundown in the Viz Mosart GUI.

For the ActiveX to function correctly, some parameters are required to be set that link the ActiveX content to your Viz Mosart Server installation.

There are two ways to link the ActiveX to your Viz Mosart Server installation:,

1. The Template Database
2. Directly to the Viz Mosart Server.

For more information, see the ActiveX section of the Viz Mosart Administrator Guide earlier than version 5.5.

13.10 Installing the Viz Mosart Database

This section presents both installing and creating a database for Viz Mosart templates.

- MySQL Database Servers and Clients
- MySQL Users
- Initial Mosart Template Database
  - File Location
  - Creating an Initial Template Database

Prerequisites

- A MySQL or equivalent, database server (MariaDB, AWS Aurora, MySQL) installed and accessible from the Viz Mosart server.
  This database server can be either dedicated to the Mosart automation servers or be an existing in-house or cloud-based server.

> **Note**: We recommend considering *MariaDB* as it is an open-source platform that still provides robust database management capabilities and compatibility with MySQL standards.

- A database user with credentials sufficient for creating and altering the database. This is also mandatory for AV Automation, which automatically updates the template database.
- A local MySQL client or tool that can connect and run SQL scripts towards the database.

> **Note**: The Viz Mosart server can operate with *both* MySQL **5.x** and **8**.x.

13.10.1 MySQL Database Servers and Clients

- Wampserver: A complete Windows web development environment using Apache2 and PHP. It contains both MySQL and MariaDB database servers and supports PhpMyAdmin or Adminer for database management and configuration.

  > **Note**: Wampserver contains MySQL 5.x

  This method is recommended when installing a dedicated template database for the Mosart automation servers. It provides remote access to the database, via http.
- MySQL server: Use the MySQL Installer to install the last native MySQL database server. Recommended to also install the MySQL Workbench as a client for managing and configuring the database server.
- MySQL Workbench: Provides an integrated tools environment for database management and administration.
- MariaDB: Open source equivalent of MySQL server.
- HeidiSQL: Simple free MySql/ProgreSQL/Microsoft SQL client.
- dbVisualizer: Client for multiple database servers.

Refer to section Prerequisites for details of supported versions.

Verify the connection to the template database

Before configuring the Mosart server, verify the *connection* between the template database and the Mosart server, using a MySql client.
Typical issues at this stage relate to firewall and credentials/permissions.

13.10.2 MySQL Users

There are four types of users of a Mosart template database:

Role	Description	Application
Server management	Full rights, at least the possibility to create and alter databases	
Database management	Full rights within a single database / schema	Minimum rights for AV Automation
Database use (read/write)	Needs CRUD operations on a single database	Minimum rights for Manus Admin
Database use (read only)	Only need read operations on a single database	Minimum rights for ActiveX

It is recommend is to create *two* users for Mosart automation.
- One for the Mosart Server (AV Automation and ManusAdmin) will full admin rights on the Mosart database.
- One for the Mosart ActiveX, which only requires rights to perform read operations.

13.10.3 Initial Mosart Template Database

File Location

The initial database is located in the *Sql* directory as part of the Mosart server installation.

- Pick the *.sql* file with the highest version number.
  In this example, 1.0.5:

```
InitialMosartTemplateDatabase_1.0.5.sql
```

This SQL file (containing the initial Mosart Template Database) is used to create the initial database on the database server.

Creating an Initial Template Database

To create a Template Database

You can create the initial template database in one of two ways:

1. By importing the initial database SQL file using a database client. All MySql database clients given above supports import of SQL scripts. Normally this requires root access.
2. Using the MySQL command line tool *mysql.exe*. The script *RunDbScripts.cmd* contains command lines to import the initial database using MySQL. This script is located in the *BatchFiles* directory as part of the Mosart server installation.

   ⚠ Option 2 (script below) requires that you have first created an empty database `mosarttemplatedb`.

   RunDbScripts.cmd

   ```
   ::: RunDbScripts.cmd:
   ::: Command script to upload an initial and empty database as the Mosart
   template database:

   ::: Option 1: Invoke mysql with user credentials on the command line.
   ::: Note localhost is assumed if host is not given.
   mysql [--host=hostname] --user=username --password=password mosarttemplatedb
   < ../Sql/InitialMosartTemplateDatabase_1.0.5.sql

   ::: Option 2: Invoke mysql with user credentials from the corresponding
   mysql.ini file.
   mysql --defaults-extra-file=mysql.ini < ../Sql/
   InitialMosartTemplateDatabase_1.0.5.sql
   ```

3. After import, a database named *mosarttemplatedb* is created.

✔ mysql.exe

mysql.exe is part of MySQL Server, MySQL Workbench or Wampserver installations

14 Maintenance

This section contains an overview of log configuration and maintenance procedures that may help streamline aspects of your broadcast environment.

This section contains the following topics:

- General Advice on System Operations
- System Logging
- Server Maintenance

14.1 General Advice on System Operations

This section contains the following topics:

- Rebooting and Restarting - General Notes
- Rebooting Machines
- Application Restart
- Viz Mosart Application Updates
- Windows Updates
- Viz Mosart Main/Backup Server Testing
- AV Automation Slow Start

14.1.1 Rebooting and Restarting - General Notes

All software system must be restarted occasionally. This applies to both the operating and Viz Mosart system.

14.1.2 Rebooting Machines

The actual reboot frequency cannot be accurately determined up front, but good practice dictates rebooting all Windows computers and servers at least once a month. Should any ill effects occur periodically even with this monthly reboot interval, it is advised to reduce the interval to once a week. Likewise, if there still are issues, the next step should be daily reboots. In this case, steps should be taken to determine the problem cause.

14.1.3 Application Restart

For restart of the Viz Mosart Applications, we have not firmly stated any interval, as this depends on the actual practice in the stations, the connected equipment and the total workflow. Just like for system reboots, good practice dictates restarting the Viz Mosart Applications at least once a month.

14.1.4 Viz Mosart Application Updates

Vizrt regularly provides feature and maintenance updates for Viz Mosart system applications. Please review the associated Release Notes for each release to help with assessing whether you should upgrade.
After an upgrade, we always recommend performing some test runs in between other transmission schedules, focused on your mission-critical workflows.

However, if your operation is running according to your expectations, and is not dependent on the published fixes of a specific update, or as long as you don't require the new functionality that the update introduces, or when your servers are running in network isolation, then as long as your system has a valid maintenance contract with Vizrt Support, there may not be a need to perform an upgrade of your Viz Mosart system.

14.1.5 Windows Updates

The interval for running Windows updates depends on the environment. If the computers are accessible from other zones, both Windows and antivirus updates must be done just as any other machine in the station’s network. Based on the possibility of antivirus runaway, it is not recommended to run real-time virus scanning during transmissions. 24/7 stations must give this possible issue thorough considerations.

Copyright © 2026 Vizrt

14.1.6 Viz Mosart Main/Backup Server Testing

|Our recommended usage pattern is to alternate between main and backup every day, to make sure that the vital backup server is ready to take over the day you really need it. This way, you are continuously testing the backup function.

14.1.7 AV Automation Slow Start

There can be an approximately 30 second delay when starting AV Automation. This delay is either caused by

1. The Viz Mosart SNMP service being enabled but with incorrect SNMP connection settings.
2. A pending Windows update, queued to run.
   In this case, follow the appropriate Microsoft procedure to update Windows, using downloaded files (when you are not permitted to connect to the Internet).

Copyright © 2026 Vizrt

14.2 System Logging

All logging in Viz Mosart is done through the Log Service. This service typically runs on the server hosting the Viz Mosart server and accessed through Windows Services.

All applications transmit log events to the server, including the Viz Mosart GUI. When the Viz Mosart GUI is running on a different machine than the Viz Mosart server (the normal case), a selected subset of local log messages are transferred, via the Remote Control Service, to the Viz Mosart server. This enables these messages to also be logged in the Viz Mosart server logs.

There are three distinct log streams:

- The Technical Log: Standard logging from all applications.
- The As-run Log: Information about on-air events.
- The MOS Log: Information about MOS communication between a MOS-based NRCS and Viz Mosart's Manus Administrator.

Note: For setups where *several* Mosart applications are installed on the same machine, using their own logging service (for example, the Viz Mosart UI, Viz Mosart's Audio Player and the Viz Mosart server), if logging services do not provide the expected results, please ensure that the Viz Mosart server was the *last* component to be installed and that the Mosart Logging Service points to the Mosart server installation folder. This is particularly relevant when one of the *log adapters* supported by Mosart is used. See section Log Adapters below.

This section contains the following topics:

- Log Viewer
- Log File Characteristics
  - Location
  - Format
  - File Naming
  - Current and Historical
- Log Content
  - Technical Log
  - As-run Log
  - MOS Log
- Log Adapters
- Configuring Logging
- Starting-up a Log Adapter
- Troubleshooting Log Adapters
  - Verifying Log Adapter Performance

14.2.1 Log Viewer

A Viz Mosart installation includes a dedicated log file viewer, *Log Viewer*. This utility is included in the Viz Mosart server installation directory. It is also available from Vizrt Support as a separate installation package.

Tip: To improve filtering and visualization, the XML log files are readable in Microsoft Excel.

14.2.2 Log File Characteristics

Location

Viz Mosart logs' default directory is:

- *C:\MMLogs*
  The Technical log stream is located at the root of this directory. The two other log streams have their own corresponding subdirectories.
  The location is configurable.

Format

All Viz Mosart logs are stored as XML files.

File Naming

Log file names are configurable, but typically follow the pattern:

```
{0}.{1:yyyyMMdd-hhmmss}.xml
```

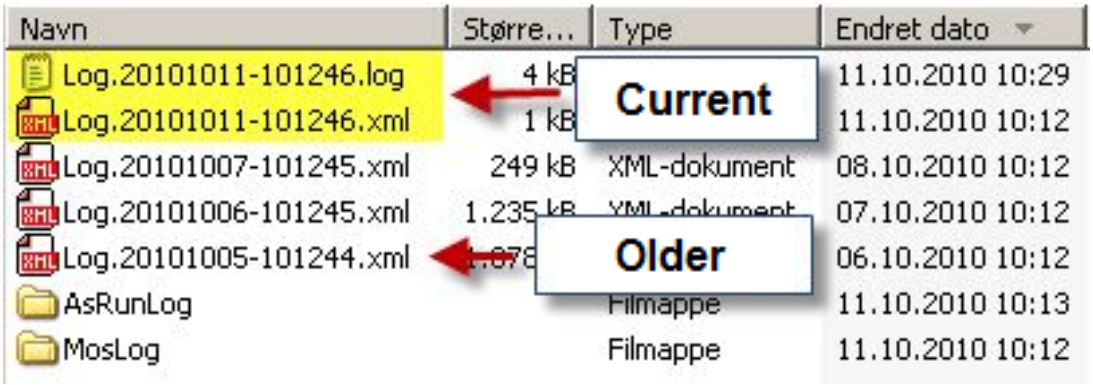
Where:

- { `0` }: User configurable prefix. Default: *Log*, *AsRunLog* and *MosLog* for the three logs respectively.
- { `1` }: File creation time, in the form *year-month-day hour-minutes-seconds*.

Current and Historical

There are two sets of log files:

- The *current*, active log file.
  - For efficiency purposes, this log is divided into two separate files, an XML file and a log file.
  - When reading the current file it is recommended to open the *XML*, as it contains a reference to the corresponding log file.
- *Historical* log files.
  - When a new log file starts, the current active log is merged into a single XML. The figure below illustrates a default setup, with a list of log files belonging to the Technical log and two directories, each containing similar log files for As-run and MOS logs respectively.



14.2.3 Log Content

Technical Log

This log contains standard logging events, produced when running Viz Mosart Applications.

> **Info:** This logging includes any Viz Mosart application normally run on other host machines, but belonging to the same control room.

Event types

All events in the Technical log stream are classified according to these event types:

- **AsRunLogger:** See As-run Log.
- **AudioMixer:** Events to/from the Audio Mixer.
- **AudioPlayer:** Events from the Audio Player and messages from AV Automation to Audio Player.
- **AvAutomation:** General events issued from AV Automation.
- **ConsoleController:** Used by default by GenericController. For generic console applications (services).
- **Crossover:** Events to/from the Crossover component.
- **Database:** Events associated with the SNMP Service.
- **GeneralInfo:** General events not assigned to any device or application.
- **GPIControl:** Events to/from the GPI control.
- **Graphics:** Events related to control of graphics.
- **GUI:** Events from the Viz Mosart GUI, such as mouse and button events.
- **GUILocal:** Messages to the local log service from the Viz Mosart GUI. This is an extended set of local GUI events, logged to the local GUI client. It excludes pressed keys, these are sent to the Server log.
- **INewsWebServices:** Events associated with iNews web service.
- **IntegratedEngine:** Events to/from the integrated Engine.
- **Instrumentation:** Events related to the Instrumentation component in AV Automation.
- **LightControl:** Events to/from the Light control.
- **Loudness:** Events to/from the Loudness control.
- **ManusAdministrator:** General events issued from Manus Administrator.
- **MediaAdministrator:** General events issued from Media Administrator.
- **MediaAssetManagement:** Used principally for the Amadeus component.

Copyright © 2026 Vizrt

- **MediaRouter:** Media Router events.
- **Mimic:** Events associated with Mimic functionality.
- **MosartRemote:** Events related to remote control of Viz Mosart.
- **MosConnection:** MOS events between MOS based NRCS and Viz Mosart.
- **RoboticCameraControl:** Events to/from the Robotic camera control.
- **RouterControl:** Events to/from the Router control.
- **SoundFilePlayer:** Events to/from the Sound File Player.
- **SpeakNoticer:** Events associated with the Asio Speak Notifier (for Mimic).
- **Tally:** Events to/from the Tally control.
- **TemplateSharing:** Template sharing events.
- **Texting:** Events to/from the subtitles control.
- **TimeDisplay:** Events associated with Timing Info display.
- **Timing:** For Timing Display logging.
- **UserMessage:** User messages from Control Commands.
- **VideoServer:** Events to/from the Video Server controller.
- **VideoSwitcher:** Events to/from the Video Switcher controller.
- **VideoWallController:** Events to/from the Video Wall controller.
- **VirtualSet:** Principally for the BrainstormVS component.
- **WeatherControl:** Events to/from the Weather controller.

GUI verbose logging

Similar to other Viz Mosart applications, the Viz Mosart GUI can be configured to perform verbose logging of application events. Verbose logging is only intended for troubleshooting, so by default it is turned *off*.

> **Note**: When the Viz Mosart GUI is running on a different machine than the Viz Mosart server (which is the normal case), a selected subset of local log messages are transferred, via the Remote Control Service, to the Viz Mosart server, enabling these messages to be logged in the Viz Mosart server logs.
> **It is important to note** that when the Viz Mosart GUI runs on a *separate* machine, filtering of verbose log messages ***will not, by default work as intended***.
> The unwanted effect is that *all* verbose log messages are transferred to the Viz Mosart server, even if verbose logging is turned *off* in the Viz Mosart GUI.
>
> - This issue was addressed in Viz Mosart 5.5, but as the setting is local and preserved at upgrade, any earlier (not a clean installation of 5.5) version of Viz Mosart
> **requires a configuration change** in the Viz Mosart GUI configuration file
> *%localappdata%/Mosart_Medialab/MosartMultiGui.exe_.../user.config*,
> **changing "VerboseLevel" from the default "*0*" to "*-1*"**.

All Viz Mosart Applications have a set of configuration properties to control what events are passed to the Log Service:

- **Verbose**:
  - Turns on/off verbose logging.
- **VerboseIgnoreEvents**:
  - A comma separated list of log event types to ignore during verbose logging.
  - Default: *<empty>* (no events are ignored).
- **VerbosePassEvents**:

- A comma separated list of log event types that are passed during verbose logging.
- Default: *<empty>* (all events are passed).

> **Info:** Setting `VerbosePassEvents = VideoServer,AudioMixer` ensures that only verbose events to/from video servers and audio mixer are passed to the Log Service.

As-run Log

This log stream contains events associated with content that has been on-air, typically all video and graphic content. The Log Service may be configured to extract information from the As-run log, for input into media asset management or presentation automation systems. All events in the As-run log stream are classified according to theie event type:

- **CROSSPOINT:** Issued as response to a crosspoint change from a tally feedback.
- **LOG:** Issued at start and stop of logging to the As-run log stream.
- **NEXT_STORY:** Issued when the *Next story* element for on-air is changed.
- **RUNDOWN_RELOAD:** Issued when the rundown is reloaded.
- **STORY_ENDED:** Issued when the current story is ended.
- **STORY_STARTED:** Issued when the current story is started.
- **TAKE_EXTERNALS:** Issued when externals are taken on air.
- **TAKE_ITEM:** Issued when a story item is taken on air.

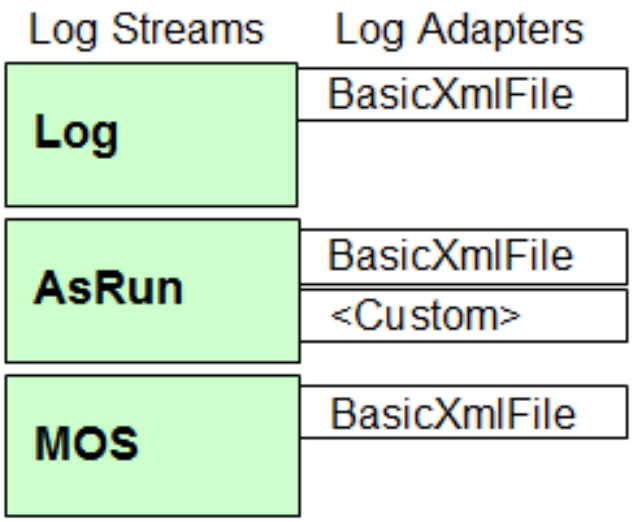
> **Info:** A customized As-run log adapter uses unique syntax.

MOS Log

This log stream contains events communicated between a MOS-based NRCS and Viz Mosart's Manus Administrator.

14.2.4 Log Adapters

Each log stream may be processed simultaneously by a set of *log adapters* as illustrated in the figure below:



In this example, three log streams are configured:

- The Technical Log Stream has one adapter:
  - **BasicXmlFile**: A default adapter that creates XML-based log files.
- The AsRun Log Stream has two adapters:
  - **BasicXmlFile**: A default adapter that creates XML-based log files.
  - **Custom adapter**: Processes the As-run log stream according to site specific requirements. These adapters have to be custom built and are not included in the standard Viz Mosart installer.
- The MOS Log Stream has one adapter:
  - **BasicXmlFile**: As above.

14.2.5 Configuring Logging

You can configure log content, granularity and naming.

- Configuration Files
- Configuration of Log Adapters
- Configuration of Log Filters
- Customizing an As-run Log Adapter
- AsRun Log Publisher

Configuration Files

Each log has its own configuration file, enabling adjustment of log properties like filename, or how often a new log file is generated.
The default log configuration files are located in the directory `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\ConfigurationFiles`

If changes are needed to any of the default settings, the applicable configuration file must first be copied to the folder `%ProgramData%\Mosart Medialab\ConfigurationFiles`
before applying any changes.

Log configuration file naming:

- **Technical log**: *LogRepositoryConfig.xml*.
- **MOS log:** *MosLogRepositoryConfig.xml*.
  This log file comprises three segments:
  - **Adapters**: Configuration of one or more log adapters.
  - **FilterBank**: Configuration of optional log filters.
  - **Properties**: Configuration of log properties.
- **As-run log**: *AsRunLogRepositoryConfig.xml*.

Example of an As-run log

A sample log configuration file for the As-run log stream:

```
<?xml version=”1.0” encoding=“utf-8“ ?>
<LogRepositoryConfig>
<!- No specified adapters shall result in default adapter -->
<Adapters>
<!- Default adapter -->
<Adapter type=“BasicXmlFileAdapter“ name=“AsRunLog“ />
<!- Custom adapter -->
<Adapter type=“LogAdapterDRMorpheus“ name=“DRMorpheus“
configuration=“DRMorpheusAsRunLogConfig.xml“ assembly=“LogAdapterDRMorpheus.dll“ />
</Adapters>
<!-- Empty filter bank shall result in logging of all events -->
<FilterBank>
<LogFilter event=“TAKE_ITEM“ pattern=““ action=“Log“ />
<LogFilter event=“LOG“ pattern=““ action=“Ignore“ />
<LogFilter event=“LOG“ pattern=“.*break.*“ action=“Break“ />
</FilterBank>
<Properties>
<item name=“AdapterType“ value=“BasicXmlFileAdapter“ />
<item name=“RepositoryPath“ value=“C:\MMLogs\AsRunLog“ />
<item name=“FilePrefix“ value=“AsRunLog“ />
<item name=“MaxDaysInRepository“ value=“60“ />
<item name=“MaxFileNumberOfEvents“ value=“10000“ />
<item name=“MaxFileTimePeriodHours“ value=“24“ />
</Properties>
</LogRepositoryConfig>
```

Tip: For customer specific as-run log files, see section Customizing an As-run Log Adapter below.

Configuration file properties

All configuration files have the same set of properties:

- **AdapterType:** The default log adapter. Default: *BasicXmlFileAdapter*.
- **RepositoryPath:** The directory where the log files are stored. Default: *C:\MMLogs*, *C:\MMLogs\AsRunLog*, *C:\MMLogs\MosLog*.

- **FilePrefix:** Custom file prefix for the log file names. Default: *Log*, *AsRunLog*, *MosLog*.
- **MaxDaysInRepository:** Maximum number of days before the logfile is automatically deleted. Default: *60*.
- **MaxFileNumberOfEvents:** Maximum number of log events before a new logfile is created. Default: *10000*.
- **MaxFileTimePeriodHours:** Maximum number of hours before a new logfile is created. Default: *24*.
- **UseLocalTime:** If set to *true*, all timestamps are converted from UTC to local time before being passed to the log adapter. Default: *False*.

Configuration of Log Adapters

A log stream may have one, or several, log adapters. These are configured in the *Adapters* section as shown in Example of an As-run Log above.
If the *Adapters* section is omitted, or empty, a single log adapter of the default type is created.

> **Important**: When initially configuring a new log adapter, after an installation or upgrade, you must complete all configurations before installing and starting the Viz Mosart Server.
> If this sequence is not observed, the log adapter will not perform as expected. See Troubleshooting Log Adapters.

The default log adapter type is configurable in the *Properties* section, but is normally set to `BasicXmlFileAdapter`.

A single adapter has the following attributes:

- **type:** The adapter type.
- **name:** Name used for identifying the log activities associated with the adapter.
- **configuration:** Optional configuration file for the log adapter. If no configuration file is given the log adapter inherits the values from the current configuration file.
- **assembly:** Optional .Net assembly (.dll) containing the log adapter. This enables dynamic log adapters, loaded at run-time.

Configuration of Log Filters

A log adapter may have a set of filters that remove log events before they are handled by the adapter.

If no filters are defined, or the filter bank is empty, all events pass to the event handling log adapter.

Each filter has three attributes:

- **event:** Which event the filter applies to. Events are listed in sections Technical Log and AsRun Log. Setting this attribute to *DEFAULT* changes the settings for the default filter bank. The default filter is used for all events that do not have a designated filter bank.
- **pattern:** Optional regular expression pattern that is applied to the log event value. The filter only returns the corresponding action when the pattern returns a match. If no pattern is given, the corresponding action is returned for all events.
- **action:** The action to return. The following actions are available:
    - **Ignore:** The event is ignored.
    - **Log:** The event is logged (forwarded to the log adapter).
    - **Break:** The event is treated as a Break event.

Example of filtering a Technical log stream

```
<!- Empty filter bank, passes all log events -->
<FilterBank />
<!- Only passes events of type “MosConnection“ -->
<FilterBank>
<LogFilter event=“DEFAULT“ action=“Ignore“ />
<LogFilter event=“MosConnection“ action=“Log“ />
</FilterBank>
<!- Passes all log events, but treats “InstantCollect“ events and all Events
containing the word “break“ as a break event -->
<FilterBank>
<LogFilter event=“InstantCollect“ pattern=“” action=“Break“ />
<LogFilter event=“DEFAULT“ pattern=“.*break.*“ action=“Break“ />
</FilterBank>
<!- Passes all log events, but ignores all GUI events -->
<FilterBank>
<LogFilter event=“Gui“ pattern=“” action=“Ignore“ />
</FilterBank>
```

Customizing an As-run Log Adapter

Customizing is performed on a site-specific basis, by Vizrt staff. Some samples are presented below for reference. Customization is based on the standard *AsRunLogRepositoryConfig.xml* as presented in Example of an As-run Log, and then include two more configuration files that can contain elements that override this standard configuration file.

- *xxxAsRunLogConfig.xml*, (Overrides AsRunlogRepositoryConfig)
- *xxxSchedulerConfig.xml*, (Overrides xxxAsRunLogConfig)

Info: xxx is here substituted for the custom naming.

Customization example: *xxxAsRunLogConfig.xml*

```
<?xml version=”1.0“ encoding=“utf-8“ ?>
<LogRepositoryConfig>
<FilterBank>
<!-- Triggers all “NEXT_BREAK“ events as breaks... -->
<LogFilter event=“NEXT_BREAK“ pattern=“” action=“Break“ />
<LogFilter event=“STORY_STARTED“ pattern=“BREAK“ action=“Break“ />
<LogFilter event=“STORY_ENDED“ pattern=“BREAK“ action=“Break“ />
</FilterBank>
<Properties>
<!-- Special configuration for the xxx log adapter -->
<item name=“SchedulerConfig“ value=“xxxSchedulerConfig.xml“ />
<!-- These values may be overridden in SchedulerConfig -->
<item name=“RepositoryPath“ value=“C:\MMLOGS\AsRunLog\xxx“ />
```

```
<item name="FilePrefix" value="UPD" />
<!-- Set to true to use local time codes instead of UTC -->
<item name="UseLocalTime" value="True" />
<!-- Clean up command for files. Set to 0 if no cleanup -->
<item name="MaxDaysInRepository" value="0" />
</Properties>
</LogRepositoryConfig>
```

Customization example: *xxxSchedulerConfig.xml*

```
<?xml version="1.0" encoding="utf-8" ?>
<SchedulerConfig>
<ShowPrefix>xxx</ShowPrefix>
<FilePattern>{Name}_{Prefix}_{Date}_{DayOfWeek}_{Intime}_{Rundown}_{StoryTitle}
_{StoryId}_{StoryIndex}</FilePattern>
<Directory>C:\MMLOGS\AsRunLog\xxx</Directory>
<Extension>xml</Extension>
<DayOfWeek>Dim,Lun,Mar,Mer,Jeu,Ven,Sam</DayOfWeek>
<!--<DayOfWeek>Dimanche,Lundi,Mardi,Mercredi,Jeudi,Vendredi,Samedi</DayOfWeek> -->
<NoLoggingDuringRehearsal>true</NoLoggingDuringRehearsal>  <!-- If false, logging are
permitted during rehearsal -->
<UseAbsoluteTime>true</UseAbsoluteTime> <!-- If false, time codes are relative to
show start -->
<MinLogDelay>-1</MinLogDelay> <!-- Specifies a minimum logging delay after configured
show start (in seconds) -->
<OneFilePerStory>true</OneFilePerStory> <!-- True if a file is to be generated per
story -->
<FlushAtStoryEnd>false</FlushAtStoryEnd> <!-- True if file should be flushed when
story ends. Default when show ends. -->
<IgnoreEmptyShows>true</IgnoreEmptyShows> <!-- True if file should be flushed when
story ends. Default when show ends. -->
</SchedulerConfig>
```

AsRun Log Publisher

The AsRun Log Publisher enables Viz Mosart to forward *As-run* log events to an external REST-based service.

To enable or customize an AsRun Log Publisher

This example enables forwarding of logs using HTTP POST.

1. From your Mosart installations' ConfigurationFiles folder, locate the file *AsRunLogRepositoryConfig.xml*.
2. Copy *AsRunLogRepositoryConfig.xml* into %PROGRAMDATA%\Mosart Medialab\ConfigurationFiles (if this file does not already exist)
3. Edit the following settings:

```
<item name="EnableNotification" value="true" />
<item name="NotificationUrl" value="http://your-rest-service:port/api" />
<item name="MaxNotificationKeepTime" value="60" />
```

```
<item name="NotificationBatchSize" value="1" />
<item name="NotificationBatchTimeOut" value="10" />
```

Batching

- **NotificationBatchSize**: The number of logs that must accumulate before sending in one batch. For example, *NotificationBatchSize=5* will hold logs until *five* items are queued, then send them all at once.
- **NotificationBatchTimeOut**: In seconds. Ensures logs are sent even if the batch size is not reached.

Retention

- **MaxNotificationKeepTime**: In minutes. How long Viz Mosart will retain logs if the REST endpoint is unreachable. Once the service is back online (within this time), the stored logs will be delivered.
  If downtime exceeds *MaxNotificationKeepTime*, unsent logs are discarded.

4. After performing these changes, restart the Mosart Log Service from Windows Services.

> **Note**: If you install the Mosart UI **after** the Mosart Server on the *same* machine, the Mosart Log Service will point to the UI folder instead of the Server folder, preventing the AsRunLog Publisher from working.
> - To fix this, install Mosart Server *after* Mosart UI or confirm that the Log Service runs from the Server installation folder.

14.2.6 Starting-up a Log Adapter

To start a log adapter

1. Modify or create your log adapter as described in the sections above.
2. Ensure that Mosart Logging Service is running on the machine where the Viz Mosart UI and Viz Mosart server are installed.
   There are two alternatives:
   a. Reinstall (UI) Mosart Logging Service or
   b. Stop the (UI) Mosart Logging Service and start the Mosart Logging Service as console, or alternatively configure to start it as as service.

14.2.7 Troubleshooting Log Adapters

Issues with log adapters usually appear in the Viz Mosart UI with errors/warnings mentioning the service Mosart Server Log *as unavailable*.

Log adapter configuration are usually made after an initial installation or upgrade that introduces a requirement for a new/changed log adapter.

- It is essential to wait with installation of the Mosart Server (from the installer bundle) till the log adapter configuration is complete, and changed files saved in the `%ProgramData%\Mosart Medialab\ConfigurationFiles` folder.
  If you did not observe this installation sequence, simply uninstall and reinstall Mosart Server.

Verifying Log Adapter Performance

To verifying Log Adapter performance

If you have unexpected logging service events, you can check the status of a log adapter.

1. Stop *Mosart Logging Service*.
2. Ensure that the affected logging configuration files, that initially reside in the `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\` folder
   - *AsRunLogRepositoryConfig.xml*
   - *LogAdapterxxxConfig.xml* (here *xxx* is the naming of your new log adapter)
   already exist in *either*:
   - under `%ProgramData%\Mosart Medialab\ConfigurationFiles` folder
   or
   - `C:\ChannelTemplates` folder
   (but *never* in both places! Remove the oldest, and place the remaining one in the preferred location, `%ProgramData%\Mosart Medialab\ConfigurationFiles`).
3. Ensure that in *AsRunLogRepositoryConfig.xml,* the section corresponding to your log adapter (for example `<!-- Adapter for Viz One -- ...`)
   is *uncommented* around the config parameters.
4. Save the file if you made changes.
5. Open your log adapter config file (*LogAdapterxxxConfig.xml*) and ensure the configurations match your system and save, if you made any changes.
6. In `%ProgramFiles(x86)%\Mosart Medialab\Mosart Server\`, start the Mosart as-run service, `MMAsRunLogService.exe` as a *console*.
7. In the console, check that your log adapter initialized, and that it has successfully read the settings from the configuration file.
   There is plenty of useful information in this readout.
   Example:

```
12:26:59 I   3  1 MMLog              LogStatus        Logging started: Trace only
12:26:59 I   3  1 MainLogger         MMAsRunLogService version=5.0.0.32224 Beta build=5.0.0.32224
12:26:59 I   3  1 MainLogger         MMLogService     Instansiated Mosart Log Service
12:26:59 I   3  1 AsRunLogger        Initialize       Initializes log service of type AsRunLogServer
12:26:59 I   3  1 AsRunLogger        Initialize       Configuration file: C:\ChannelTemplates\AsRunLogRepositoryConfig.xml
12:27:00 I   3  1 AsRunLogger        Initialize       Creating log repository of type AdapterContainer
12:27:00 I   3  1 AsRunLog           XML File Log repository initialization:
12:27:00 I   3  1 AsRunLog           RepositoryPath=C:\MMLogs\AsRunLog
12:27:00 I   3  1 AsRunLog           MaxFileNumberOfEvents=10000
12:27:00 I   3  1 AsRunLog           MaxFileTimePeriodHours=2
12:27:00 I   3  1 AsRunLog           MaxDaysInRepository=60
12:27:00 I   3  1 AsRunLogger        Added log repository of type[BasicXmlFileAdapter]: AsRunLog
12:27:00 I   3  7 AsRunLog           Merges all local logfiles from repository
12:27:00 I   3  7 AsRunLog           Log file created: AsRunLog.20210930-142700.xml
12:27:00 I   3  7 AsRunLog           Next file age check @ 9/30/2021 4:27:10 PM
12:27:00 I   3  7 AsRunLog           Removes all logfiles from repository older than 60 days
12:27:00 I   3  1 LogAdapterStoryRecorder Initialize       Initializing the log adapter
12:27:00 I   3  7 AsRunLog           Next file repository check @ 10/1/2021 12:00:00 AM
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     PrerollTemplate: PREROLL
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     EdlWatchFolder: \\bgo-mos-server1\EDL\
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     EdlLocalFolder: C:\MMLogs\EDL
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     RecordingFolder: \\10.[redacted]34\fs0\clip.dir
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     RecordingClipNamePrefix: SRVizrt
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     RecordingRetentionDays: 10
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     HandleGetSoMDelay: 2 seconds
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     HandleTakeItemDelay: 2000 seconds
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     SnippetGenerationRequestDelay: 15 seconds
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     SnippetLengthForEachRecording: 00:00:05:00 seconds
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     StoryRecorderSnippetClipsFolder: \\bgo-mos-server1\StoryRecorderSnippe
12:27:00 I   3  1 LogAdapterStoryRecorder ReadProperty     EnableSrSessionLogging: True
12:27:00 I   3  1 LogAdapterStoryRecorder StartListening   Backup server listening at: 0.0.0.0:5001
12:27:00 I   3  1 LogAdapterStoryRecorder StartListening   Listening to backup server at: bgo-mos-server1:5001
12:27:00 I   3  1 LogAdapterStoryRecorder Stop             Stopped scheduled cleanup process for recordings
12:27:00 I   3  1 LogAdapterStoryRecorder Start            Scheduling cleanup process for recordings
12:27:00 I   3  1 LogAdapterStoryRecorder Initialize       Log adapter initialized
12:27:00 I   3  1 AsRunLogger        Added log repository of type[LogAdapterStoryRecorder]: LogAdapterStoryRecorder
12:27:00 I   3 20 LogAdapterStoryRecorder OnInitialize     Requesting SERVER_IDLE_STATE.
12:27:00 I   3  1 AsRunLogger        Initialize       Verifies service endpoint: net.tcp://localhost:8091/AsRunLog/
12:27:00 I   3  1 AsRunLogger        Initialize       Verifies service endpoint: net.tcp://localhost:8091/AsRunLog/mex
12:27:00 I   3  1 MainLogger         MMLogService     Instansiated Mosart Log Service
12:27:00 I   3  1 MainLogger         Initialize       Initializes log service of type LogServer
12:27:00 I   3  1 MainLogger         Initialize       Configuration file: C:\ChannelTemplates\LogRepositoryConfig.xml
12:27:00 I   3  1 MainLogger         Initialize       Creating log repository of type BasicXmlFileAdapter
12:27:00 I   3  1 MainLogger         XML File Log repository initialization:
12:27:00 I   3  1 MainLogger         RepositoryPath=C:\MMLogs
```

Start-up details

Values successfully read

Running OK

8. After this verification step, *close* the console.
9. Start `MMAsRunLogService` as a service.

14.3 Server Maintenance

This section describes various maintenance aspects of Viz Mosart, and is divided into the following:

- Server File Structure
  - Static Configurations
  - User Configurations
  - Viz Mosart Applications and Services
  - Manus Administrator Repository
  - Log Directory
- Files for Backup
- File Purging

14.3.1 Server File Structure

The Viz Mosart Server stores files in different locations on the workstation as indicated by the list below:

Location	Definition
C:\Program Files (x86)\Mosart Medialab	Mosart applications. Various sub-directories for components such as Mosart GUI, Mosart Server and more. Either *\Program Files* or *\Program Files (x86)*.
C:\channeltemplates	XML-files defining the channel templates. Normally contains at a minimum *ChannelTemplates.xml*, *newsroomsettings.xml*, *avconfig.xml*.
C:\manus	The Run-down (playlist) files fetched from the Newsroom systems.
C:\MMLogs	Default location for Log-files. Rotated.
C:\ProgramData\Mosart Medialab\ConfigurationFiles	System-wide configuration settings (XML).
C:\Users\<USERNAMEHERE>\AppData\Local\Mosart_Medialab	User-specific application configuration files.

Note: The list above is an example and may differ for various installations and Windows versions. See below for further details.

Static Configurations

This is configuration that is static and common to all versions of Viz Mosart installed on the computer. Any upgrades do not overwrite any the content mentioned below but new configuration parameters are appended if present.

- **Location:** *C:\channeltemplates*.
- **Type:** Configuration.
- **Backup:** Yes.
- **For Support:** Yes.
- **Content:**
    - Video and audio setup.
    - Story templates.

User Configurations

This is configuration that is dependent upon the location and version of the application. Typically under the location there is a set of directories each referring to a particular installation of a Viz Mosart application. For example, for the Media Administrator:

- *%UserProfile%\Local Settings\Application Data\Mosart_Medialab\MMConsoleAdmin_2007.exe_Url_<*>\<version number>.* Any upgrades create a new subdirectory name with the corresponding version number and do a merge from the current version.
- **Location:** *%AppData%\Mosart_Medialab* or *%UserProfile%\Local Settings\Application Data\Mosart_Medialab* (i.e. for a particular non-roaming user: *C:\Documents and Settings\<User>\Local Settings\Application Data\Mosart_Medialab*)
- **Type:** Configuration.
- **Backup:** Yes.
- **For Support:** Yes.
- **Content:** Settings for all Viz Mosart Applications, for example the content of the various settings dialogs.

Viz Mosart Applications and Services

This is the location where the Viz Mosart installers place program files.

- **Location:** *%ProgramFiles(x86)%/Mosart Medialab* (i.e. for a default English language setup: *C:\Program Files(x86)\Mosart Medialab*).
- **Type:** Applications.
- **Backup:** Yes* - Most of the executables (.exe) files have corresponding configuration files. Normally the content in there is static and is not changed during an upgrade. Nevertheless, custom configuration is possible.
- **For Support:** No.
- **Content:** One or several subdirectories. One for each Viz Mosart installer.

Copyright © 2026 Vizrt

Manus Administrator Repository

The Manus Administrator repository contains all rundowns that have been loaded into Viz Mosart from the newsroom system. All rundowns are stored in an XML format, making it possible to extract rundown information to use for other purposes, for example video clip usage statistics.

- **Location:** *C:\manus*.
- **Type:** Rundowns.
- **Backup:** No* - Only for selected rundowns.
- **For Support:** Yes.
- **Content:** One or several subdirectories.

Log Directory

By default, log files are stored in *C:\MMLogs*.

- **Location:** *C:\MMLogs*
- **Type:** Log.
- **Backup:** No.
- **For Support:** Yes.
- **Content:** A set of log files from the various Viz Mosart Applications.

Note: For support purposes, collect all log files with timestamps that cover the period of time where an issue was discovered.

14.3.2 Files for Backup

All locations stated in Server File Structure should have a safety copy taken whenever changes are made to the active Viz Mosart Server. It is recommended to copy files and corresponding directories to a common share, outside Main and Backup Viz Mosart Servers, marked with Viz Mosart version and dated when the backup was performed.

14.3.3 File Purging

The *Manus Administrator Repository* and the *Log Directory* above, have built in mechanisms for purging files. The content of these directories grows unless a maintenance procedure is established.

For the *Manus Administrator Repository*, the retention period can be configured in the Manus Administrator Configuration by writing `settings` in the Manus Administrator console window, and then setting the value *ManusExpirationTime* to any number of days. The default value for *ManusExpirationTime* is 60 (days). If the manus files have an in-house maintenance procedure in place, use -1 and Manus Administrator will not delete any files. 0 is not a legal value. Only xml files auto-generated by Manus Administrator will be deleted.

The *Log Directory* system is preset to a 60 days retention period, but this can be changed, see System Logging.

15 Appendix

This section covers the following topics:

- GPIO with Sample XML Config
- Structure of MOS Objects Sent to NRCS
- Syntax of Textual Definitions
- Vizrt SSO (Keycloak)

15.1 GPIO with Sample XML Config

Note: This example relates to the topic GPI/O.

15.1.1 Sample Configuration (W&T Web-IO)

W&T's Web-IO products enable TCP/IP Ethernet-based switching signals for control, acquisition and monitoring. Their products are accessed using API queries from, for example, a standard web browser, or in this case, Viz Mosart.

From the Web-IO side, GPI is enabled by attaching a device capable of sending GPI signals to Viz Mosart. The configuration below shows configuration of a W&T Web-IO device capable of 24 GPI/GPO connections.
This configuration uses at a maximum 24 GPO signals. A maximum of 12 GPI signals can be used to control Viz Mosart.

GPI 0-4

The first 5 GPI signals are connected to fixed Viz Mosart commands for backward compatibility.

The signals are as follows:
GPI 0 - Reload rundown. Triggers a rundown reload event. The timeline will be stopped and the current rundown will be reloaded from the NCS cache.
GPI 1 - Start/continue rundown. Triggers the TAKE-NEXT event and with then start the timeline or advance to the next timeline element.
GPI 2 - Start rundown from top. Starts the rundown from the first story.
GPI 3 - Disable rehearsal mode.
GPI 4 - Enable rehearsal mode.

GPI 5-11

Each of the GPI signals 5-11 may be assigned to either trigger a Viz Mosart control command, trigger an arbitrary template from the active template set or trigger a direct take.

```
Working configuration from Viz Mosart Lab in Bergen, October 2014
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
V2.30, Aug  1 2013, 12:55:23
-->
<io-Digital12x12.3>
    <Config>
        <Device>
            <Text>
                <DeviceName>WEBIO-<wut1></DeviceName>
                <DeviceText> From the terminal block directly to the network </
DeviceText>
                <Location></Location>
                <Contact></Contact>
            </Text>
```

```
<Time_Date>
    <TimeZone>
        <UTCoffset>01:00</UTCoffset>
        <Enable>
            <Apply_Time_Zone>ON</Apply_Time_Zone>
        </Enable>
        <Summertime>
            <UTCoffset>02:00</UTCoffset>
            <Enable>
                <Apply_Summertime>ON</Apply_Summertime>
            </Enable>
            <Start>
                <Month>
                    <January>OFF</January>
                    <February>OFF</February>
                    <March>ON</March>
                    <April>OFF</April>
                    <May>OFF</May>
                    <June>OFF</June>
                    <July>OFF</July>
                    <August>OFF</August>
                    <September>OFF</September>
                    <October>OFF</October>
                    <November>OFF</November>
                    <December>OFF</December>
                </Month>
                <Mode>
                    <first>OFF</first>
                    <second>OFF</second>
                    <third>OFF</third>
                    <fourth>OFF</fourth>
                    <last>ON</last>
                </Mode>
                <Weekday>
                    <Sunday>ON</Sunday>
                    <Monday>OFF</Monday>
                    <Tuesday>OFF</Tuesday>
                    <Wednesday>OFF</Wednesday>
                    <Thursday>OFF</Thursday>
                    <Friday>OFF</Friday>
                    <Saturday>OFF</Saturday>
                </Weekday>
                <Time>02:00</Time>
            </Start>
            <Stop>
                <Month>
                    <January>OFF</January>
                    <February>OFF</February>
                    <March>OFF</March>
                    <April>OFF</April>
                    <May>OFF</May>
                    <June>OFF</June>
                    <July>OFF</July>
```

```
                    <August>OFF</August>
                    <September>OFF</September>
                    <October>ON</October>
                    <November>OFF</November>
                    <December>OFF</December>
                </Month>
                <Mode>
                    <first>OFF</first>
                    <second>OFF</second>
                    <third>OFF</third>
                    <fourth>OFF</fourth>
                    <last>ON</last>
                </Mode>
                <Weekday>
                    <Sunday>ON</Sunday>
                    <Monday>OFF</Monday>
                    <Tuesday>OFF</Tuesday>
                    <Wednesday>OFF</Wednesday>
                    <Thursday>OFF</Thursday>
                    <Friday>OFF</Friday>
                    <Saturday>OFF</Saturday>
                </Weekday>
                <Time>03:00</Time>
            </Stop>
        </Summertime>
    </TimeZone>
    <TimeServer>
        <UTCServer1>de.pool.ntp.org</UTCServer1>
        <UTCServer2>europe.pool.ntp.org</UTCServer2>
        <SyncTime>0</SyncTime>
        <Enable>
            <Apply_TimeServer>OFF</Apply_TimeServer>
        </Enable>
    </TimeServer>
    <DeviceClock>
        <Time>15:56</Time>
        <Day>01</Day>
        <Month>01</Month>
        <Year>2004</Year>
    </DeviceClock>
</Time_Date>
<BasicSettings>
    <Network>
        <IP_Addr>10.211.114.124</IP_Addr>
        <Subnet_Mask>255.255.255.00</Subnet_Mask>
        <Gateway>10.211.114.01</Gateway>
        <BOOTPClient>
            <STATIC>ON</STATIC>
            <BOOTP_enable>OFF</BOOTP_enable>
            <DHCP_enable>OFF</DHCP_enable>
        </BOOTPClient>
        <DnsServer1></DnsServer1>
        <DnsServer2></DnsServer2>
```

```
        <KeepAliveTime>00</KeepAliveTime>
    </Network>
    <HTTP_Set>
        <Startup>
            <index.htm>ON</index.htm>
            <home.htm>OFF</home.htm>
            <user.htm>OFF</user.htm>
        </Startup>
        <Enable>
            <GET_Header_enable>OFF</GET_Header_enable>
            <GET_HTTP_enable>OFF</GET_HTTP_enable>
        </Enable>
        <_HTTP_Port>80</_HTTP_Port>
        <InputTrigger>
            <Input_0>ON</Input_0>
            <Input_1>ON</Input_1>
            <Input_2>ON</Input_2>
            <Input_3>ON</Input_3>
            <Input_4>ON</Input_4>
            <Input_5>ON</Input_5>
            <Input_6>ON</Input_6>
            <Input_7>ON</Input_7>
            <Input_8>ON</Input_8>
            <Input_9>ON</Input_9>
            <Input_10>ON</Input_10>
            <Input_11>ON</Input_11>
        </InputTrigger>
    </HTTP_Set>
    <HTTP_Client>
        <HTTP_Client>
            <HTTP_Client_enable>OFF</HTTP_Client_enable>
        </HTTP_Client>
        <Enable>
            <GET_Header_enable>OFF</GET_Header_enable>
            <GET_HTTP_enable>OFF</GET_HTTP_enable>
        </Enable>
        <LocalPort>AUTO</LocalPort>
        <ServerPort></ServerPort>
        <Server_IP_Addr></Server_IP_Addr>
        <ServerUrl></ServerUrl>
        <InactiveTimeout></InactiveTimeout>
        <InputTrigger>
            <Input_0>OFF</Input_0>
            <Input_1>OFF</Input_1>
            <Input_2>OFF</Input_2>
            <Input_3>OFF</Input_3>
            <Input_4>OFF</Input_4>
            <Input_5>OFF</Input_5>
            <Input_6>OFF</Input_6>
            <Input_7>OFF</Input_7>
            <Input_8>OFF</Input_8>
            <Input_9>OFF</Input_9>
            <Input_10>OFF</Input_10>
```

```
            <Input_11>OFF</Input_11>
        </InputTrigger>
        <Timer_Cron></Timer_Cron>
    </HTTP_Client>
    <Mail>
        <Name>Web-IO 12xDigital</Name>
        <ReplyAddr>Web-IO@no.reply</ReplyAddr>
        <MailServer></MailServer>
        <Authentication>
            <_SMTP_authentication_off>ON</_SMTP_authentication_off>
            <_ESMTP>OFF</_ESMTP>
            <_SMTP_after_POP3>OFF</_SMTP_after_POP3>
        </Authentication>
        <User></User>
        <Password></Password>
        <POP3_Server></POP3_Server>
        <Enable>
            <Mail_enable>OFF</Mail_enable>
        </Enable>
    </Mail>
    <SNMP>
        <Community_String_Read></Community_String_Read>
        <Community_String_Read_Write></Community_String_Read_Write>
        <Manager_IP></Manager_IP>
        <SystemTraps>
            <Cold_Start>OFF</Cold_Start>
            <Warm_Start>OFF</Warm_Start>
            <Diag_Messages>OFF</Diag_Messages>
        </SystemTraps>
        <Enable>
            <SNMP_enable>ON</SNMP_enable>
        </Enable>
    </SNMP>
    <UDP>
        <Port>42279</Port>
        <RemotePort></RemotePort>
        <Enable>
            <UDP_enable>OFF</UDP_enable>
        </Enable>
    </UDP>
    <Syslog>
        <SyslogServerIP></SyslogServerIP>
        <SyslogServerPort>514</SyslogServerPort>
        <SystemMessages>
            <Cold_Start>OFF</Cold_Start>
            <Warm_Start>OFF</Warm_Start>
            <Diag_Messages>OFF</Diag_Messages>
        </SystemMessages>
        <Enable>
            <System_Messages_enable>OFF</System_Messages_enable>
        </Enable>
    </Syslog>
    <FTP>
```

```
            <FtpServerIP></FtpServerIP>
            <FtpControlPort>21</FtpControlPort>
            <User></User>
            <Password></Password>
            <FtpAccount></FtpAccount>
            <Options>
                <PASV>OFF</PASV>
            </Options>
            <Enable>
                <FTP_enable>OFF</FTP_enable>
            </Enable>
        </FTP>
        <Binary1>
            <OperationMode>
                <TCP_Server>ON</TCP_Server>
                <TCP_Client>OFF</TCP_Client>
                <UDP_Peer>OFF</UDP_Peer>
            </OperationMode>
            <Enable>
                <Enable_BINARY_1>ON</Enable_BINARY_1>
            </Enable>
            <TcpServer1>
                <LocalPort>49153</LocalPort>
                <Client_HTTP_Port>80</Client_HTTP_Port>
                <InputTrigger>
                    <Input_0>ON</Input_0>
                    <Input_1>ON</Input_1>
                    <Input_2>ON</Input_2>
                    <Input_3>ON</Input_3>
                    <Input_4>ON</Input_4>
                    <Input_5>ON</Input_5>
                    <Input_6>ON</Input_6>
                    <Input_7>ON</Input_7>
                    <Input_8>ON</Input_8>
                    <Input_9>ON</Input_9>
                    <Input_10>ON</Input_10>
                    <Input_11>ON</Input_11>
                </InputTrigger>
                <ApplicationMode1>
                    <Socket_device>ON</Socket_device>
                    <Compatible_50xxx>OFF</Compatible_50xxx>
                    <Box2Box_Slave>OFF</Box2Box_Slave>
                    <OPC_Device>OFF</OPC_Device>
                </ApplicationMode1>
            </TcpServer1>
            <TcpClient1>
                <LocalPort>AUTO</LocalPort>
                <ServerPort>49153</ServerPort>
                <Server_HTTP_Port>80</Server_HTTP_Port>
                <Server_IP_Addr></Server_IP_Addr>
                <ServerPassword></ServerPassword>
                <InactiveTimeout>30</InactiveTimeout>
                <InputTrigger>
```

```
                <Input_0>ON</Input_0>
                <Input_1>ON</Input_1>
                <Input_2>ON</Input_2>
                <Input_3>ON</Input_3>
                <Input_4>ON</Input_4>
                <Input_5>ON</Input_5>
                <Input_6>ON</Input_6>
                <Input_7>ON</Input_7>
                <Input_8>ON</Input_8>
                <Input_9>ON</Input_9>
                <Input_10>ON</Input_10>
                <Input_11>ON</Input_11>
            </InputTrigger>
            <Interval></Interval>
            <ApplicationMode2>
                <Socket_device>ON</Socket_device>
                <Compatible_50xxx>OFF</Compatible_50xxx>
                <Box2Box_Master>OFF</Box2Box_Master>
                <ILOface>OFF</ILOface>
            </ApplicationMode2>
        </TcpClient1>
        <UdpPeer1>
            <LocalPort>45889</LocalPort>
            <RemotePort>45889</RemotePort>
            <Remote_IP_Addr></Remote_IP_Addr>
            <InputTrigger>
                <Input_0>ON</Input_0>
                <Input_1>ON</Input_1>
                <Input_2>ON</Input_2>
                <Input_3>ON</Input_3>
                <Input_4>ON</Input_4>
                <Input_5>ON</Input_5>
                <Input_6>ON</Input_6>
                <Input_7>ON</Input_7>
                <Input_8>ON</Input_8>
                <Input_9>ON</Input_9>
                <Input_10>ON</Input_10>
                <Input_11>ON</Input_11>
            </InputTrigger>
            <Interval></Interval>
            <ApplicationMode3>
                <Socket_device>ON</Socket_device>
                <Compatible_50xxx>OFF</Compatible_50xxx>
            </ApplicationMode3>
        </UdpPeer1>
    </Binary1>
    <Binary2>
        <OperationMode>
            <TCP_Server>ON</TCP_Server>
            <TCP_Client>OFF</TCP_Client>
            <UDP_Peer>OFF</UDP_Peer>
        </OperationMode>
        <Enable>
```

```
        <Enable_BINARY_2>ON</Enable_BINARY_2>
    </Enable>
    <TcpServer2>
        <LocalPort>49154</LocalPort>
        <Client_HTTP_Port>80</Client_HTTP_Port>
        <InputTrigger>
            <Input_0>ON</Input_0>
            <Input_1>ON</Input_1>
            <Input_2>ON</Input_2>
            <Input_3>ON</Input_3>
            <Input_4>ON</Input_4>
            <Input_5>ON</Input_5>
            <Input_6>ON</Input_6>
            <Input_7>ON</Input_7>
            <Input_8>ON</Input_8>
            <Input_9>ON</Input_9>
            <Input_10>ON</Input_10>
            <Input_11>ON</Input_11>
        </InputTrigger>
        <ApplicationMode1>
            <Socket_device>ON</Socket_device>
            <Compatible_50xxx>OFF</Compatible_50xxx>
            <Box2Box_Slave>OFF</Box2Box_Slave>
            <OPC_Device>OFF</OPC_Device>
        </ApplicationMode1>
    </TcpServer2>
    <TcpClient2>
        <LocalPort>AUTO</LocalPort>
        <ServerPort>49154</ServerPort>
        <Server_HTTP_Port>80</Server_HTTP_Port>
        <Server_IP_Addr></Server_IP_Addr>
        <ServerPassword></ServerPassword>
        <InactiveTimeout>30</InactiveTimeout>
        <InputTrigger>
            <Input_0>ON</Input_0>
            <Input_1>ON</Input_1>
            <Input_2>ON</Input_2>
            <Input_3>ON</Input_3>
            <Input_4>ON</Input_4>
            <Input_5>ON</Input_5>
            <Input_6>ON</Input_6>
            <Input_7>ON</Input_7>
            <Input_8>ON</Input_8>
            <Input_9>ON</Input_9>
            <Input_10>ON</Input_10>
            <Input_11>ON</Input_11>
        </InputTrigger>
        <Interval></Interval>
        <ApplicationMode2>
            <Socket_device>ON</Socket_device>
            <Compatible_50xxx>OFF</Compatible_50xxx>
            <Box2Box_Master>OFF</Box2Box_Master>
            <ILOface>OFF</ILOface>
```

```
                </ApplicationMode2>
            </TcpClient2>
            <UdpPeer2>
                <LocalPort>45890</LocalPort>
                <RemotePort>45890</RemotePort>
                <Remote_IP_Addr></Remote_IP_Addr>
                <InputTrigger>
                    <Input_0>ON</Input_0>
                    <Input_1>ON</Input_1>
                    <Input_2>ON</Input_2>
                    <Input_3>ON</Input_3>
                    <Input_4>ON</Input_4>
                    <Input_5>ON</Input_5>
                    <Input_6>ON</Input_6>
                    <Input_7>ON</Input_7>
                    <Input_8>ON</Input_8>
                    <Input_9>ON</Input_9>
                    <Input_10>ON</Input_10>
                    <Input_11>ON</Input_11>
                </InputTrigger>
                <Interval></Interval>
                <ApplicationMode3>
                    <Socket_device>ON</Socket_device>
                    <Compatible_50xxx>OFF</Compatible_50xxx>
                </ApplicationMode3>
            </UdpPeer2>
        </Binary2>
    </BasicSettings>
    <OutputMode>
        <Output0>
            <_HTTP>ON</_HTTP>
            <_UDP_ASCII>OFF</_UDP_ASCII>
            <_SNMP>OFF</_SNMP>
            <Binary_1>OFF</Binary_1>
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output0>
        <Output1>
            <_HTTP>ON</_HTTP>
            <_UDP_ASCII>OFF</_UDP_ASCII>
            <_SNMP>OFF</_SNMP>
            <Binary_1>OFF</Binary_1>
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output1>
        <Output2>
            <_HTTP>ON</_HTTP>
            <_UDP_ASCII>OFF</_UDP_ASCII>
            <_SNMP>OFF</_SNMP>
            <Binary_1>OFF</Binary_1>
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output2>
```

```
<Output3>
    <_HTTP>ON</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>OFF</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output3>
<Output4>
    <_HTTP>ON</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>OFF</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output4>
<Output5>
    <_HTTP>ON</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>OFF</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output5>
<Output6>
    <_HTTP>OFF</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>ON</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output6>
<Output7>
    <_HTTP>OFF</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>ON</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output7>
<Output8>
    <_HTTP>OFF</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>ON</Binary_1>
    <Binary_2>OFF</Binary_2>
    <_Logic>OFF</_Logic>
</Output8>
<Output9>
    <_HTTP>OFF</_HTTP>
    <_UDP_ASCII>OFF</_UDP_ASCII>
    <_SNMP>OFF</_SNMP>
    <Binary_1>ON</Binary_1>
```

```
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output9>
        <Output10>
            <_HTTP>OFF</_HTTP>
            <_UDP_ASCII>OFF</_UDP_ASCII>
            <_SNMP>OFF</_SNMP>
            <Binary_1>ON</Binary_1>
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output10>
        <Output11>
            <_HTTP>OFF</_HTTP>
            <_UDP_ASCII>OFF</_UDP_ASCII>
            <_SNMP>OFF</_SNMP>
            <Binary_1>ON</Binary_1>
            <Binary_2>OFF</Binary_2>
            <_Logic>OFF</_Logic>
        </Output11>
        <Safety_Timeout></Safety_Timeout>
        <Enable>
            <Load_Control_enable>OFF</Load_Control_enable>
        </Enable>
    </OutputMode>
    <Alarm>
        <Alarm1>
            <InputTrigger>
                <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
                <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
                <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
                <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
                <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
                <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
                <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
                <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
            </InputTrigger>
            <OutputTrigger>
                <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
```

```
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
                        <FtpLocalDataPort>AUTO</FtpLocalDataPort>
                        <FileName></FileName>
```

```
                        <FtpAlarmText></FtpAlarmText>
                        <Alarm_clear_text></Alarm_clear_text>
                        <Options>
                            <STORE>OFF</STORE>
                            <APPEND>ON</APPEND>
                        </Options>
                    </FTP>
                </Alarm1>
                <Alarm2>
                    <InputTrigger>
                        <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
                        <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
                        <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
                        <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
                        <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
                        <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
                        <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
                        <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                        <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                        <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                        <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                        <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
                    </InputTrigger>
                    <OutputTrigger>
                        <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                        <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                        <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                        <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                        <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
```

```
            <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
            <FTP_Client_enable>OFF</FTP_Client_enable>
        </AlarmEnable>
        <Mail>
            <E-Mail-Addr></E-Mail-Addr>
            <Subject></Subject>
            <Mailtext></Mailtext>
            <Alarm_clear_subject></Alarm_clear_subject>
            <Alarm_clear_text></Alarm_clear_text>
        </Mail>
        <SNMP>
            <Manager_IP></Manager_IP>
            <TrapText></TrapText>
            <Alarm_clear_text></Alarm_clear_text>
        </SNMP>
        <UDP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <UdpText></UdpText>
            <Alarm_clear_text></Alarm_clear_text>
        </UDP>
        <TCP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <TcpText></TcpText>
            <Alarm_clear_text></Alarm_clear_text>
        </TCP>
        <Syslog>
            <IP_Addr></IP_Addr>
            <Port>514</Port>
            <SyslogText></SyslogText>
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm2>
    <Alarm3>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
```

```
                        <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                        <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                        <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                        <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                        <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
                    </InputTrigger>
                    <OutputTrigger>
                        <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                        <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                        <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                        <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                        <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
```

```
            <Port></Port>
            <UdpText></UdpText>
            <Alarm_clear_text></Alarm_clear_text>
        </UDP>
        <TCP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <TcpText></TcpText>
            <Alarm_clear_text></Alarm_clear_text>
        </TCP>
        <Syslog>
            <IP_Addr></IP_Addr>
            <Port>514</Port>
            <SyslogText></SyslogText>
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm3>
    <Alarm4>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
            <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
            <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
            <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
            <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
            <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
```

```
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
```

```
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm4>
    <Alarm5>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
            <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
            <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
            <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
            <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
            <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
            <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
            <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
        </OutputTrigger>
        <SystemTrigger>
            <Load_Control>OFF</Load_Control>
            <Interval_Timer>OFF</Interval_Timer>
            <Cold_Start>OFF</Cold_Start>
            <Warm_Start>OFF</Warm_Start>
        </SystemTrigger>
        <MaxCounterValue></MaxCounterValue>
        <Interval></Interval>
        <Timer_Cron></Timer_Cron>
        <AlarmEnable>
            <Mail_enable>OFF</Mail_enable>
            <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
            <UDP_Client_enable>OFF</UDP_Client_enable>
```

```
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
                        <FtpLocalDataPort>AUTO</FtpLocalDataPort>
                        <FileName></FileName>
                        <FtpAlarmText></FtpAlarmText>
                        <Alarm_clear_text></Alarm_clear_text>
                        <Options>
                            <STORE>OFF</STORE>
                            <APPEND>ON</APPEND>
                        </Options>
                    </FTP>
                </Alarm5>
                <Alarm6>
                    <InputTrigger>
                        <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
                        <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
                        <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
                        <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
```

```
                        <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
                        <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
                        <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
                        <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                        <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                        <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                        <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                        <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
                    </InputTrigger>
                    <OutputTrigger>
                        <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                        <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                        <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                        <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                        <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
```

```
        </SNMP>
        <UDP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <UdpText></UdpText>
            <Alarm_clear_text></Alarm_clear_text>
        </UDP>
        <TCP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <TcpText></TcpText>
            <Alarm_clear_text></Alarm_clear_text>
        </TCP>
        <Syslog>
            <IP_Addr></IP_Addr>
            <Port>514</Port>
            <SyslogText></SyslogText>
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm6>
    <Alarm7>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
            <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
            <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
```

```
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
```

```
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm7>
    <Alarm8>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
            <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
            <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
            <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
            <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
            <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
            <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
            <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
        </OutputTrigger>
        <SystemTrigger>
            <Load_Control>OFF</Load_Control>
            <Interval_Timer>OFF</Interval_Timer>
            <Cold_Start>OFF</Cold_Start>
            <Warm_Start>OFF</Warm_Start>
        </SystemTrigger>
        <MaxCounterValue></MaxCounterValue>
        <Interval></Interval>
        <Timer_Cron></Timer_Cron>
        <AlarmEnable>
            <Mail_enable>OFF</Mail_enable>
```

```
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
                        <FtpLocalDataPort>AUTO</FtpLocalDataPort>
                        <FileName></FileName>
                        <FtpAlarmText></FtpAlarmText>
                        <Alarm_clear_text></Alarm_clear_text>
                        <Options>
                            <STORE>OFF</STORE>
                            <APPEND>ON</APPEND>
                        </Options>
                    </FTP>
                </Alarm8>
                <Alarm9>
                    <InputTrigger>
                        <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
                        <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
```

```
                        <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
                        <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
                        <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
                        <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
                        <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
                        <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                        <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                        <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                        <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                        <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
                    </InputTrigger>
                    <OutputTrigger>
                        <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                        <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                        <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                        <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                        <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
```

```
            <TrapText></TrapText>
            <Alarm_clear_text></Alarm_clear_text>
        </SNMP>
        <UDP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <UdpText></UdpText>
            <Alarm_clear_text></Alarm_clear_text>
        </UDP>
        <TCP>
            <IP_Addr></IP_Addr>
            <Port></Port>
            <TcpText></TcpText>
            <Alarm_clear_text></Alarm_clear_text>
        </TCP>
        <Syslog>
            <IP_Addr></IP_Addr>
            <Port>514</Port>
            <SyslogText></SyslogText>
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm9>
    <Alarm10>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
```

```
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
```

```
            <Port>514</Port>
            <SyslogText></SyslogText>
            <Alarm_clear_text></Alarm_clear_text>
        </Syslog>
        <FTP>
            <FtpLocalDataPort>AUTO</FtpLocalDataPort>
            <FileName></FileName>
            <FtpAlarmText></FtpAlarmText>
            <Alarm_clear_text></Alarm_clear_text>
            <Options>
                <STORE>OFF</STORE>
                <APPEND>ON</APPEND>
            </Options>
        </FTP>
    </Alarm10>
    <Alarm11>
        <InputTrigger>
            <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
            <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
            <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
            <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
            <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
            <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
            <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
            <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
            <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
            <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
            <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
            <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
        </InputTrigger>
        <OutputTrigger>
            <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
            <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
            <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
            <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
            <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
            <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
            <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
            <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
            <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
            <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
            <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
            <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
        </OutputTrigger>
        <SystemTrigger>
            <Load_Control>OFF</Load_Control>
            <Interval_Timer>OFF</Interval_Timer>
            <Cold_Start>OFF</Cold_Start>
            <Warm_Start>OFF</Warm_Start>
        </SystemTrigger>
        <MaxCounterValue></MaxCounterValue>
        <Interval></Interval>
        <Timer_Cron></Timer_Cron>
```

```
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
                        <FtpLocalDataPort>AUTO</FtpLocalDataPort>
                        <FileName></FileName>
                        <FtpAlarmText></FtpAlarmText>
                        <Alarm_clear_text></Alarm_clear_text>
                        <Options>
                            <STORE>OFF</STORE>
                            <APPEND>ON</APPEND>
                        </Options>
                    </FTP>
                </Alarm11>
                <Alarm12>
                    <InputTrigger>
```

```
                        <Input_0><enable>OFF</enable><state>OFF</state></Input_0>
                        <Input_1><enable>OFF</enable><state>OFF</state></Input_1>
                        <Input_2><enable>OFF</enable><state>OFF</state></Input_2>
                        <Input_3><enable>OFF</enable><state>OFF</state></Input_3>
                        <Input_4><enable>OFF</enable><state>OFF</state></Input_4>
                        <Input_5><enable>OFF</enable><state>OFF</state></Input_5>
                        <Input_6><enable>OFF</enable><state>OFF</state></Input_6>
                        <Input_7><enable>OFF</enable><state>OFF</state></Input_7>
                        <Input_8><enable>OFF</enable><state>OFF</state></Input_8>
                        <Input_9><enable>OFF</enable><state>OFF</state></Input_9>
                        <Input_10><enable>OFF</enable><state>OFF</state></Input_10>
                        <Input_11><enable>OFF</enable><state>OFF</state></Input_11>
                    </InputTrigger>
                    <OutputTrigger>
                        <Output_0><enable>OFF</enable><state>OFF</state></Output_0>
                        <Output_1><enable>OFF</enable><state>OFF</state></Output_1>
                        <Output_2><enable>OFF</enable><state>OFF</state></Output_2>
                        <Output_3><enable>OFF</enable><state>OFF</state></Output_3>
                        <Output_4><enable>OFF</enable><state>OFF</state></Output_4>
                        <Output_5><enable>OFF</enable><state>OFF</state></Output_5>
                        <Output_6><enable>OFF</enable><state>OFF</state></Output_6>
                        <Output_7><enable>OFF</enable><state>OFF</state></Output_7>
                        <Output_8><enable>OFF</enable><state>OFF</state></Output_8>
                        <Output_9><enable>OFF</enable><state>OFF</state></Output_9>
                        <Output_10><enable>OFF</enable><state>OFF</state></Output_10>
                        <Output_11><enable>OFF</enable><state>OFF</state></Output_11>
                    </OutputTrigger>
                    <SystemTrigger>
                        <Load_Control>OFF</Load_Control>
                        <Interval_Timer>OFF</Interval_Timer>
                        <Cold_Start>OFF</Cold_Start>
                        <Warm_Start>OFF</Warm_Start>
                    </SystemTrigger>
                    <MaxCounterValue></MaxCounterValue>
                    <Interval></Interval>
                    <Timer_Cron></Timer_Cron>
                    <AlarmEnable>
                        <Mail_enable>OFF</Mail_enable>
                        <SNMP_Trap_enable>OFF</SNMP_Trap_enable>
                        <UDP_Client_enable>OFF</UDP_Client_enable>
                        <Send_special_alarm_to_pending_TCP_connections>OFF</
Send_special_alarm_to_pending_TCP_connections>
                        <TCP_Client_enable>OFF</TCP_Client_enable>
                        <Syslog_Messages_enable>OFF</Syslog_Messages_enable>
                        <FTP_Client_enable>OFF</FTP_Client_enable>
                    </AlarmEnable>
                    <Mail>
                        <E-Mail-Addr></E-Mail-Addr>
                        <Subject></Subject>
                        <Mailtext></Mailtext>
                        <Alarm_clear_subject></Alarm_clear_subject>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Mail>
```

```
                    <SNMP>
                        <Manager_IP></Manager_IP>
                        <TrapText></TrapText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </SNMP>
                    <UDP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <UdpText></UdpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </UDP>
                    <TCP>
                        <IP_Addr></IP_Addr>
                        <Port></Port>
                        <TcpText></TcpText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </TCP>
                    <Syslog>
                        <IP_Addr></IP_Addr>
                        <Port>514</Port>
                        <SyslogText></SyslogText>
                        <Alarm_clear_text></Alarm_clear_text>
                    </Syslog>
                    <FTP>
                        <FtpLocalDataPort>AUTO</FtpLocalDataPort>
                        <FileName></FileName>
                        <FtpAlarmText></FtpAlarmText>
                        <Alarm_clear_text></Alarm_clear_text>
                        <Options>
                            <STORE>OFF</STORE>
                            <APPEND>ON</APPEND>
                        </Options>
                    </FTP>
                </Alarm12>
            </Alarm>
        </Device>
        <Ports>
            <Inputs>
                <Input0>
                    <Name>Input 0</Name>
                    <Text>Description Input 0</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Short_Pulse_detection_enable>OFF</
Short_Pulse_detection_enable>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
```

```
                            <positive>ON</positive>
                        </PulsPolarity>
                        <InactiveTimeout>1</InactiveTimeout>
                    </Counter_Mode>
                </Input0>
                <Input1>
                    <Name>Input 1</Name>
                    <Text>Description Input 1</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Short_Pulse_detection_enable>OFF</
Short_Pulse_detection_enable>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>ON</positive>
                        </PulsPolarity>
                        <InactiveTimeout>1</InactiveTimeout>
                    </Counter_Mode>
                </Input1>
                <Input2>
                    <Name>Input 2</Name>
                    <Text>Description Input 2</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Short_Pulse_detection_enable>OFF</
Short_Pulse_detection_enable>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input2>
                <Input3>
                    <Name>Input 3</Name>
                    <Text>Description Input 3</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Short_Pulse_detection_enable>OFF</
Short_Pulse_detection_enable>
                        <Change_detection_enable>OFF</Change_detection_enable>
```

```
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input3>
                <Input4>
                    <Name>Input 4</Name>
                    <Text>Description Input 4</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input4>
                <Input5>
                    <Name>Input 5</Name>
                    <Text>Description Input 5</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input5>
                <Input6>
                    <Name>Input 6</Name>
                    <Text>Description Input 6</Text>
                    <Filter></Filter>
                    <InputMode>
```

```
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input6>
                <Input7>
                    <Name>Input 7</Name>
                    <Text>Description Input 7</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input7>
                <Input8>
                    <Name>Input 8</Name>
                    <Text>Description Input 8</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input8>
                <Input9>
                    <Name>Input 9</Name>
                    <Text>Description Input 9</Text>
                    <Filter></Filter>
```

```
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input9>
                <Input10>
                    <Name>Input 10</Name>
                    <Text>Description Input 10</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input10>
                <Input11>
                    <Name>Input 11</Name>
                    <Text>Description Input 11</Text>
                    <Filter></Filter>
                    <InputMode>
                        <Change_detection_enable>OFF</Change_detection_enable>
                        <Clear_counter_on_read>OFF</Clear_counter_on_read>
                        <Bidirectional_counter_enable>OFF</
Bidirectional_counter_enable>
                    </InputMode>
                    <Counter_Mode>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                        <InactiveTimeout></InactiveTimeout>
                    </Counter_Mode>
                </Input11>
            </Inputs>
            <Outputs>
                <Output0>
```

```
                    <Name>Output 0</Name>
                    <Text>Description Output 0</Text>
                    <GroupMode>
                        <_0_1_2_3_separated>ON</_0_1_2_3_separated>
                        <_0--1_2--3_parallel>OFF</_0--1_2--3_parallel>
                        <_0--1--2--3_parallel>OFF</_0--1--2--3_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
                        <Duration>5000</Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>ON</positive>
```

```
                        </PulsPolarity>
                    </Puls>
                </Output0>
                <Output1>
                    <Name>Output 1</Name>
                    <Text>Description Output 1</Text>
                    <GroupMode>
                        <_0_1_2_3_separated>ON</_0_1_2_3_separated>
                        <_0--1_2--3_parallel>OFF</_0--1_2--3_parallel>
                        <_0--1--2--3_parallel>OFF</_0--1--2--3_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
```

```
                        <Duration>2000</Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>ON</positive>
                        </PulsPolarity>
                    </Puls>
                </Output1>
                <Output2>
                    <Name>Output 2</Name>
                    <Text>Description Output 2</Text>
                    <GroupMode>
                        <_0_1_2_3_separated>ON</_0_1_2_3_separated>
                        <_0--1_2--3_parallel>OFF</_0--1_2--3_parallel>
                        <_0--1--2--3_parallel>OFF</_0--1--2--3_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
```

```
                        <Output_Inverter>OFF</Output_Inverter>
                    </OutputInverter>
                </Logic>
                <Puls>
                    <Duration>1000</Duration>
                    <PulsPolarity>
                        <negative>OFF</negative>
                        <positive>ON</positive>
                    </PulsPolarity>
                </Puls>
            </Output2>
            <Output3>
                <Name>Output 3</Name>
                <Text>Description Output 3</Text>
                <GroupMode>
                    <_0_1_2_3_separated>ON</_0_1_2_3_separated>
                    <_0--1_2--3_parallel>OFF</_0--1_2--3_parallel>
                    <_0--1--2--3_parallel>OFF</_0--1--2--3_parallel>
                </GroupMode>
                <SafetyState>
                    <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                </SafetyState>
                <Logic>
                    <InputMask>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputMask>
                    <InputInverter>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputInverter>
                    <LogicFunc>
```

```
                        <AND>ON</AND>
                        <OR>OFF</OR>
                    </LogicFunc>
                    <OutputInverter>
                        <Output_Inverter>OFF</Output_Inverter>
                    </OutputInverter>
                </Logic>
                <Puls>
                    <Duration>1000</Duration>
                    <PulsPolarity>
                        <negative>OFF</negative>
                        <positive>ON</positive>
                    </PulsPolarity>
                </Puls>
            </Output3>
            <Output4>
                <Name>Output 4</Name>
                <Text>Description Output 4</Text>
                <GroupMode>
                    <_4_5_6_7_separated>ON</_4_5_6_7_separated>
                    <_4--5_6--7_parallel>OFF</_4--5_6--7_parallel>
                    <_4--5--6--7_parallel>OFF</_4--5--6--7_parallel>
                </GroupMode>
                <SafetyState>
                    <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                </SafetyState>
                <Logic>
                    <InputMask>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputMask>
                    <InputInverter>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
```

```
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputInverter>
                    <LogicFunc>
                        <AND>ON</AND>
                        <OR>OFF</OR>
                    </LogicFunc>
                    <OutputInverter>
                        <Output_Inverter>OFF</Output_Inverter>
                    </OutputInverter>
                </Logic>
                <Puls>
                    <Duration></Duration>
                    <PulsPolarity>
                        <negative>OFF</negative>
                        <positive>OFF</positive>
                    </PulsPolarity>
                </Puls>
            </Output4>
            <Output5>
                <Name>Output 5</Name>
                <Text>Description Output 5</Text>
                <GroupMode>
                    <_4_5_6_7_separated>ON</_4_5_6_7_separated>
                    <_4--5_6--7_parallel>OFF</_4--5_6--7_parallel>
                    <_4--5--6--7_parallel>OFF</_4--5--6--7_parallel>
                </GroupMode>
                <SafetyState>
                    <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                </SafetyState>
                <Logic>
                    <InputMask>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputMask>
                    <InputInverter>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
```

```
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputInverter>
                    <LogicFunc>
                        <AND>ON</AND>
                        <OR>OFF</OR>
                    </LogicFunc>
                    <OutputInverter>
                        <Output_Inverter>OFF</Output_Inverter>
                    </OutputInverter>
                </Logic>
                <Puls>
                    <Duration></Duration>
                    <PulsPolarity>
                        <negative>OFF</negative>
                        <positive>OFF</positive>
                    </PulsPolarity>
                </Puls>
            </Output5>
            <Output6>
                <Name>Output 6</Name>
                <Text>Description Output 6</Text>
                <GroupMode>
                    <_4_5_6_7_separated>ON</_4_5_6_7_separated>
                    <_4--5_6--7_parallel>OFF</_4--5_6--7_parallel>
                    <_4--5--6--7_parallel>OFF</_4--5--6--7_parallel>
                </GroupMode>
                <SafetyState>
                    <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                </SafetyState>
                <Logic>
                    <InputMask>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputMask>
                    <InputInverter>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
```

```
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
                        <Duration></Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                    </Puls>
                </Output6>
                <Output7>
                    <Name>Output 7</Name>
                    <Text>Description Output 7</Text>
                    <GroupMode>
                        <_4_5_6_7_separated>ON</_4_5_6_7_separated>
                        <_4--5_6--7_parallel>OFF</_4--5_6--7_parallel>
                        <_4--5--6--7_parallel>OFF</_4--5--6--7_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
```

```
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
                        <Duration></Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                    </Puls>
                </Output7>
                <Output8>
                    <Name>Output 8</Name>
                    <Text>Description Output 8</Text>
                    <GroupMode>
                        <_8_9_10_11_separated>ON</_8_9_10_11_separated>
                        <_8--9_10--11_parallel>OFF</_8--9_10--11_parallel>
                        <_8--9--10--11_parallel>OFF</_8--9--10--11_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
```

```
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputMask>
                    <InputInverter>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
                        <Input_4>OFF</Input_4>
                        <Input_5>OFF</Input_5>
                        <Input_6>OFF</Input_6>
                        <Input_7>OFF</Input_7>
                        <Input_8>OFF</Input_8>
                        <Input_9>OFF</Input_9>
                        <Input_10>OFF</Input_10>
                        <Input_11>OFF</Input_11>
                    </InputInverter>
                    <LogicFunc>
                        <AND>ON</AND>
                        <OR>OFF</OR>
                    </LogicFunc>
                    <OutputInverter>
                        <Output_Inverter>OFF</Output_Inverter>
                    </OutputInverter>
                </Logic>
                <Puls>
                    <Duration></Duration>
                    <PulsPolarity>
                        <negative>OFF</negative>
                        <positive>OFF</positive>
                    </PulsPolarity>
                </Puls>
            </Output8>
            <Output9>
                <Name>Output 9</Name>
                <Text>Description Output 9</Text>
                <GroupMode>
                    <_8_9_10_11_separated>ON</_8_9_10_11_separated>
                    <_8--9_10--11_parallel>OFF</_8--9_10--11_parallel>
                    <_8--9--10--11_parallel>OFF</_8--9--10--11_parallel>
                </GroupMode>
                <SafetyState>
                    <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                </SafetyState>
                <Logic>
                    <InputMask>
                        <Input_0>OFF</Input_0>
                        <Input_1>OFF</Input_1>
                        <Input_2>OFF</Input_2>
                        <Input_3>OFF</Input_3>
```

```
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
                        <Duration></Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                    </Puls>
                </Output9>
                <Output10>
                    <Name>Output 10</Name>
                    <Text>Description Output 10</Text>
                    <GroupMode>
                        <_8_9_10_11_separated>ON</_8_9_10_11_separated>
                        <_8--9_10--11_parallel>OFF</_8--9_10--11_parallel>
                        <_8--9--10--11_parallel>OFF</_8--9--10--11_parallel>
                    </GroupMode>
                    <SafetyState>
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
```

```
                <Input_0>OFF</Input_0>
                <Input_1>OFF</Input_1>
                <Input_2>OFF</Input_2>
                <Input_3>OFF</Input_3>
                <Input_4>OFF</Input_4>
                <Input_5>OFF</Input_5>
                <Input_6>OFF</Input_6>
                <Input_7>OFF</Input_7>
                <Input_8>OFF</Input_8>
                <Input_9>OFF</Input_9>
                <Input_10>OFF</Input_10>
                <Input_11>OFF</Input_11>
            </InputMask>
            <InputInverter>
                <Input_0>OFF</Input_0>
                <Input_1>OFF</Input_1>
                <Input_2>OFF</Input_2>
                <Input_3>OFF</Input_3>
                <Input_4>OFF</Input_4>
                <Input_5>OFF</Input_5>
                <Input_6>OFF</Input_6>
                <Input_7>OFF</Input_7>
                <Input_8>OFF</Input_8>
                <Input_9>OFF</Input_9>
                <Input_10>OFF</Input_10>
                <Input_11>OFF</Input_11>
            </InputInverter>
            <LogicFunc>
                <AND>ON</AND>
                <OR>OFF</OR>
            </LogicFunc>
            <OutputInverter>
                <Output_Inverter>OFF</Output_Inverter>
            </OutputInverter>
        </Logic>
        <Puls>
            <Duration></Duration>
            <PulsPolarity>
                <negative>OFF</negative>
                <positive>OFF</positive>
            </PulsPolarity>
        </Puls>
    </Output10>
    <Output11>
        <Name>Output 11</Name>
        <Text>Description Output 11</Text>
        <GroupMode>
            <_8_9_10_11_separated>ON</_8_9_10_11_separated>
            <_8--9_10--11_parallel>OFF</_8--9_10--11_parallel>
            <_8--9--10--11_parallel>OFF</_8--9--10--11_parallel>
        </GroupMode>
        <SafetyState>
```

```
                        <Safety_enable><enable>OFF</enable><state>OFF</state></
Safety_enable>
                    </SafetyState>
                    <Logic>
                        <InputMask>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputMask>
                        <InputInverter>
                            <Input_0>OFF</Input_0>
                            <Input_1>OFF</Input_1>
                            <Input_2>OFF</Input_2>
                            <Input_3>OFF</Input_3>
                            <Input_4>OFF</Input_4>
                            <Input_5>OFF</Input_5>
                            <Input_6>OFF</Input_6>
                            <Input_7>OFF</Input_7>
                            <Input_8>OFF</Input_8>
                            <Input_9>OFF</Input_9>
                            <Input_10>OFF</Input_10>
                            <Input_11>OFF</Input_11>
                        </InputInverter>
                        <LogicFunc>
                            <AND>ON</AND>
                            <OR>OFF</OR>
                        </LogicFunc>
                        <OutputInverter>
                            <Output_Inverter>OFF</Output_Inverter>
                        </OutputInverter>
                    </Logic>
                    <Puls>
                        <Duration></Duration>
                        <PulsPolarity>
                            <negative>OFF</negative>
                            <positive>OFF</positive>
                        </PulsPolarity>
                    </Puls>
                </Output11>
            </Outputs>
        </Ports>
        <Manufacturer>
            <Name>Wiesemann & Theis GmbH</Name>
            <Address>Porschstr. 12<br>
```

```
42279 Wuppertal<br>
Germany</Address>
            <Support_Hotline>+49-(0)202-2680-0</Support_Hotline>
            <Internet>www.wut.de</Internet>
            <DeviceTyp>Web-IO Digital 12xIn, 12xOut</DeviceTyp>
            <DeviceNo>#57630 TB 900571</DeviceNo>
            <Logo>logo.gif</Logo>
        </Manufacturer>
    </Config>
</io-Digital12x12.3>
```

15.1.2 Hardware Configuration

For this example, you configure the in and out ports as required for the GPIO system, by following their own specifications.
The Web-IO physical device can be configured via a browser pointed to the box' IP address.

To access the configuration menus

- **Config > Login > Administrator login**
  from the **Start** page.

> **Note:** Configuration changes are only saved after clicking the **Logout** button.

Mosart IAT/SAT

- For testing from Viz Mosart, the new configuration is either directly loaded, or inspected and used for setting the inport 0 manually.
  Inspection and comparison may be necessary should the load fail. If necessary reset the box to factory settings before using the configuration below.
- Under the **inport** nodes in the menu tree
  - Set the port minimum pulse filter to 100ms
  - Set the input type to positive pulse (three separate settings).
- If you use port *0* it is easier to test Viz Mosart, as this port is hardwired to the **Reload Rundown** function.

Preparation of Test Circuit

- This requires two pieces of wire, ~20cm, stripped (and preferably tinned) at the end.
- To set up the box for generating an input test pulse, interconnect the terminal strip ground connector for power supply ground (the outermost VCC connector) with the outermost GND connector, across the box. This establishes a common ground between power supply side and the input logic circuits.

> - **Note:**
  - It is not possible to simply take the output of one output port and feed it directly back to the input side, as the outputs are intended to drive *relay circuits* (these are also known as *current drivers,* they operate at a very low voltage)
  - The **12 in 12 out** version requires an *external* relay.

- The inputs requires 8V referred to GND to detect a pulse.

Testing GPI 0

- To generate the input pulse, connect a piece of wire from the other VCC connector and briefly touch the connector for Inport 0 (the pulse must be >100ms).
- A green light beside the connector will briefly light up if successfully triggered. Given that AV Automation has GPIO enabled and is also enabled for Input 0, any loaded rundown will reload, clearly visible on the Mosart GUI.
- Configure GPIO with the file containing the working configuration.

15.2 Structure of MOS Objects Sent to NRCS

You can send Mosart templates as MOS objects to an NRCS and then use them to create items in a Viz Mosart rundown (as an alternative to the Mosart ActiveX method) as a brick system.
In addition to MOS objects that contain Viz Mosart templates, there is a special MOS object that can be sent containing lower thirds information.

This topic is covered in section Mosart Templates Feedback to the NRCS.

This section provides some sample MOS object files:

- MOS Object with Mosart Templates Grouped by Type
- MOS Object with Mosart Templates not Grouped by Type
- MOS Object with Clip Templates Merged in a Mosart Template
- Structure of the MOS object with Lowerthird Information
- Octopus

15.2.1 MOS Object with Mosart Templates Grouped by Type

- Variants for Mosart Templates of Same Type
- Variants for Mosart Templates of Same Type and SendAllTemplateSets set to True

Variants for Mosart Templates of Same Type

A single MOS object will be created for Mosart templates of same type from the default template set. In this example

- <mosExternalMetadata> node contains the list of variants for the Mosart templates, of same type.
- <objID> of the MOS object is the template type, for example 0 for CAMERA or 1 for PACKAGE.
- <objGroup> is the default template set name.
- <type> node has the name set to the template type name (for example CAMERA, PACKAGE etc.), and *templateset* set to the name of the default template set.

```
<mosObj>
    <objID>0</objID>
    <objSlug>CAMERA</objSlug>
    <mosAbstract>CAMERA</mosAbstract>
    <objGroup>Talkshow</objGroup>
    <objType>STILL</objType>
    <objTB>50</objTB>
    <objRev>1</objRev>
    <objDur>0</objDur>
    <status>NEW</status>
    <objAir>NOT READY</objAir>
    <createdBy>
    </createdBy>
    <created>2019-03-04T13:23:38,000Z</created>
    <changedBy>
    </changedBy>
    <changed>2020-03-07T09:22:21,256Z</changed>
    <description>CAMERA</description>
    <mosExternalMetadata>
    <mosScope>PLAYLIST</mosScope>
    <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
    <mosPayload>
        <mosarttemplate>
        <type name="CAMERA" templateset=”Talkshow” category="">
            <variants fieldtype="LIST" value="1">
            <variant name="1">
                <transitions value="DEFAULT" enable="false">
                <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                </transition>
                <transition name="MIX">
                    <field name="transitionrate" value="0" fieldtype="NUMBER"
range="0,999" />
                </transition>
```

```
                <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
range="0,999" />
                </transition>
                <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
keylist="effects" />
                </transition>
                </transitions>
                <fields />
            </variant>
            <variant name="2">
                <transitions value="DEFAULT" enable="false">
                <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                </transition>
                <transition name="MIX">
                    <field name="transitionrate" value="0" fieldtype="NUMBER"
range="0,999" />
                </transition>
                <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
range="0,999" />
                </transition>
                <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
keylist="effects" />
                </transition>
                </transitions>
                <fields />
            </variant>
            </variants>
            <keys name="mixerinputs">
            <k>
            </k>
            <k>CAM1</k>
            <k>CAM2</k>
            <k>GRAPHIC1</k>
            <k>GRAPHIC2</k>
            <k>LIVE1</k>
            <k>LIVE2</k>
            <k>SKYPE1</k>
            <k>SKYPE2</k>
            <k>PLAYA</k>
            <k>PLAYB</k>
            <k>PLAYC</k>
            <k>PLAYD</k>
            <k>ME1</k>
            <k>ME2</k>
            <k>ME3</k>
            </keys>
            <keys name="effects">
            <k>1</k>
```

```
            <k>2</k>
            <k>3</k>
            <k>4</k>
            <k>5</k>
            <k>6</k>
            <k>7</k>
            <k>8</k>
            <k>9</k>
            <k>10</k>
            <k>11</k>
            <k>12</k>
            <k>13</k>
            <k>14</k>
            <k>45</k>
            <k>47</k>
            </keys>
            <keys name="routersources">
            <k>SAT_10</k>
            <k>DEP_01_A</k>
            </keys>
            <keys name="routerdestinations">
            <k>M_VISU_2</k>
            <k>M_ROBO_2</k>
            <k>TESTLOW</k>
            <k>TESTHIGH</k>
            </keys>
        </type>
        </mosarttemplate>
    </mosPayload>
    </mosExternalMetadata>
  </mosObj>
```

Variants for Mosart Templates of Same Type and SendAllTemplateSets set to True

A single MOS object will be created for Mosart templates of the same type for each template set. In this example:

- <mosExternalMetadata> node contains the list of variants for the Mosart templates of the same type.
  For example, if there are 3 template sets and 10 template types having 2 variants each, then there are *30* MOS objects created, each having 2 variants.
- <objID> is the “TemplateSet-TemplateType”.
  For example in the ’FuncSpec’ template set, FuncSpec-0 for CAMERA or FuncSpec-1 for PACKAGE.
- <objSlug> is the template type name.
  For example, CAMERA or PACKAGE.
- <objGroup> is the template set name.
- <type> node has the name of the template type name (for example CAMERA, PACKAGE etc.), and *templateset* set to the name of the template set.
  In the example below there are there are 2 template sets:
  - *FuncSpec* having 3 Camera templates and
  - *Demo* having 2 Camera templates.

```
<mosObj>
    <objID>FUNCSPEC-0</objID>
    <objSlug>CAMERA</objSlug>
    <mosAbstract>CAMERA</mosAbstract>
    <objGroup>Funcspec</objGroup>
    <objType>STILL</objType>
    <objTB>50</objTB>
    <objRev>1</objRev>
    <objDur>0</objDur>
    <status>NEW</status>
    <objAir>NOT READY</objAir>
    <createdBy>
    </createdBy>
    <created>2016-04-01T11:42:09,000Z</created>
    <changedBy>
    </changedBy>
    <changed>2017-09-22T12:05:20,070Z</changed>
    <description>CAMERA</description>
    <mosExternalMetadata>
      <mosScope>PLAYLIST</mosScope>
      <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
      <mosPayload>
        <mosarttemplate>
          <type name="CAMERA" templateset="Funcspec" category="">
            <variants fieldtype="LIST" value="LIVE">
              <variant name="LIVE">
                <transitions value="DEFAULT" enable="false">
                  <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                  </transition>
                  <transition name="MIX">
                    <field name="transitionrate" value="0" fieldtype="NUMBER"
 range="0,999" />
                  </transition>
                  <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
 range="0,999" />
                  </transition>
                  <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
 keylist="effects" />
                  </transition>
                </transitions>
                <fields>
                  <field name="xpoint1" value="CAM1" default="CAM1" fieldtype="LIST"
 keylist="mixerinputs" />
                </fields>
              </variant>
              <variant name="21">
                <transitions value="DEFAULT" enable="false">
                  <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
```

```
                  </transition>
                  <transition name="MIX">
                    <field name="transitionrate" value="2" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
keylist="effects" />
                  </transition>
                </transitions>
                <fields />
              </variant>
              <variant name="20">
                <transitions value="DEFAULT" enable="false">
                  <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                  </transition>
                  <transition name="MIX">
                    <field name="transitionrate" value="2" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
keylist="effects" />
                  </transition>
                </transitions>
                <fields />
              </variant>
            </variants>
            <keys name="mixerinputs">
              <k>
              </k>
              <k>CAM1</k>
              <k>CAM2</k>
              <k>LIVE1</k>
              <k>VIZ</k>
              <k>IN11_HD1</k>
              <k>IN12_HD2</k>
              <k>VIZ2</k>
              <k>ME1</k>
              <k>ME2</k>
              <k>LIVE2</k>
              <k>DUMMY</k>
              <k>DDR1</k>
              <k>DDR2</k>
```

```
            </keys>
            <keys name="effects">
              <k>1</k>
              <k>2</k>
              <k>3</k>
              <k>4</k>
              <k>5</k>
              <k>6</k>
              <k>7</k>
              <k>8</k>
              <k>9</k>
            </keys>
            <keys name="routersources">
              <k>R CAM 1</k>
            </keys>
            <keys name="routerdestinations">
              <k>OPUS 1</k>
            </keys>
          </type>
        </mosarttemplate>
      </mosPayload>
    </mosExternalMetadata>
  </mosObj>
<mosObj>
    <objID>DEMO-0</objID>
    <objSlug>CAMERA</objSlug>
    <mosAbstract>CAMERA</mosAbstract>
    <objGroup>DEMO</objGroup>
    <objType>STILL</objType>
    <objTB>50</objTB>
    <objRev>1</objRev>
    <objDur>0</objDur>
    <status>NEW</status>
    <objAir>NOT READY</objAir>
    <createdBy>
    </createdBy>
    <created>2014-11-11T12:55:05,000Z</created>
    <changedBy>
    </changedBy>
    <changed>2016-05-04T11:01:27,903Z</changed>
    <description>CAMERA</description>
    <mosExternalMetadata>
      <mosScope>PLAYLIST</mosScope>
      <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
      <mosPayload>
        <mosarttemplate>
          <type name="CAMERA" templateset="DEMO" category="">
            <variants fieldtype="LIST" value="1">
              <variant name="1">
                <transitions value="DEFAULT" enable="false">
                  <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                  </transition>
```

```
                  <transition name="MIX">
                    <field name="transitionrate" value="0" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="WIPE">
                    <field name="transitionrate" value="10" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="EFFECT">
                    <field name="effectname" value="0" fieldtype="LIST"
keylist="effects" />
                  </transition>
                </transitions>
                <fields />
              </variant>
               <variant name="2">
                <transitions value="DEFAULT" enable="false">
                  <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                  </transition>
                  <transition name="MIX">
                    <field name="transitionrate" value="50" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="WIPE">
                    <field name="transitionrate" value="50" fieldtype="NUMBER"
range="0,999" />
                  </transition>
                  <transition name="EFFECT">
                    <field name="effectname" value="0" fieldtype="LIST"
keylist="effects" />
                  </transition>
                </transitions>
                <fields />
              </variant>
            </variants>
            <keys name="mixerinputs">
              <k>
              </k>
              <k>CAM1</k>
              <k>CAM2</k>
              <k>LIVE1</k>
              <k>VIZ</k>
              <k>IN11_HD1</k>
              <k>IN12_HD2</k>
              <k>VIZ2</k>
              <k>ME1</k>
              <k>ME2</k>
              <k>LIVE2</k>
              <k>DUMMY</k>
              <k>DDR1</k>
              <k>DDR2</k>
            </keys>
```

```
            <keys name="effects">
              <k>1</k>
              <k>2</k>
              <k>3</k>
              <k>4</k>
              <k>5</k>
              <k>6</k>
              <k>7</k>
              <k>8</k>
              <k>9</k>
            </keys>
            <keys name="routersources">
              <k>R CAM 1</k>
            </keys>
            <keys name="routerdestinations">
              <k>OPUS 1</k>
            </keys>
          </type>
        </mosarttemplate>
      </mosPayload>
    </mosExternalMetadata>
  </mosObj>
```

15.2.2 MOS Object with Mosart Templates not Grouped by Type

For each Mosart template, a MOS object will be created to be sent to NCS. In this example:

- <mosExternalMetadata>is the node containing the variant of the Mosart template it represents.
- <objID> is the format <TemplateSet>-<Type>-<Variant>.
- <objSlug> has <Type> <Variant>;<Name>.
- <objGroup> is set to the template set the template belongs to.
- <type> node has its name set to the type of the template.
  For example CAMERA, PACKAGE etc, and *templateset* has the name of the template set this template belongs to.
- Other information is:
  - The list of video crosspoints (“mixerinputs”)
  - Effect numbers (“effects”)
  - Router sources (“routersources”)
  - Router destinations (“routerdestinations”)
    as configured in AvConfig.xml file.

The name of these key lists must be exactly as given in the parentheses, because they can be referenced from a Mosart item when adding newsroomtags to a template.

```
<mosObj>
    <objID>Talkshow-1-FULL</objID>
    <objSlug>PACKAGE FULL;name...</objSlug>
    <mosAbstract>PACKAGE FULL;description</mosAbstract>
    <objGroup>Talkshow</objGroup>
    <objType>STILL</objType>
    <objTB>50</objTB>
    <objRev>1</objRev>
    <objDur>0</objDur>
    <status>UPDATED</status>
    <objAir>NOT READY</objAir>
    <createdBy/>
    <created/>
    <changedBy/>
    <changed/>
    <description>PACKAGE</description>
    <mosExternalMetadata>
    <mosScope>PLAYLIST</mosScope>
    <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
    <mosPayload>
        <mosarttemplate>
        <type name="PACKAGE" templateset=”Talkshow” category="CLIP">
            <variants fieldtype="LIST" value="FULL">
            <variant name="FULL">
                <transitions value="DEFAULT" enable="false">
                <transition name="DEFAULT">
                    <field name="" value="" fieldtype="TEXT" />
                </transition>
```

```
                <transition name="MIX">
                    <field name="transitionrate" value="0" fieldtype="NUMBER"
range="0,999" />
                </transition>
                <transition name="WIPE">
                    <field name="transitionrate" value="" fieldtype="NUMBER"
range="0,999" />
                </transition>
                <transition name="EFFECT">
                    <field name="effectname" value="" fieldtype="LIST"
keylist="effects" />
                </transition>
                </transitions>
                <fields>
                <field name="clip_hirespath" fieldtype="TEXT" />
                <field name="clip_description" fieldtype="TEXT" />
                <field name="clip_mark_in" default="00:00:00:00"
     fieldtype="TIMECODE" inputmask="hh:mm:ss:ff" />
                <field name="clip_dur" default="00:00:00"
         fieldtype="TIMECODE" inputmask="mm:ss:ff" />
                </fields>
            </variant>
            </variants>
            <keys name="mixerinputs">
            <k>
            </k>
            <k>CAM1</k>
            <k>CAM2</k>
            <k>GRAPHIC1</k>
            <k>GRAPHIC2</k>
            <k>LIVE1</k>
            <k>LIVE2</k>
            <k>SKYPE1</k>
            <k>SKYPE2</k>
            <k>PLAYA</k>
            <k>PLAYB</k>
            <k>PLAYC</k>
            <k>PLAYD</k>
            <k>ME1</k>
            <k>ME2</k>
            <k>ME3</k>
            </keys>
            <keys name="effects">
            <k>1</k>
            <k>2</k>
            <k>3</k>
            <k>4</k>
            <k>5</k>
            <k>6</k>
            <k>7</k>
            <k>8</k>
            <k>9</k>
            <k>10</k>
```

```
            <k>11</k>
            <k>12</k>
            <k>13</k>
            <k>14</k>
            <k>45</k>
            <k>47</k>
            </keys>
            <keys name="routersources">
            <k>SAT_10</k>
            <k>DEP_01_A</k>
            </keys>
            <keys name="routerdestinations">
            <k>M_VISU_2</k>
            <k>M_ROBO_2</k>
            <k>TESTLOW</k>
            <k>TESTHIGH</k>
            </keys>
        </type>
        </mosarttemplate>
    </mosPayload>
    </mosExternalMetadata>
  </mosObj>
```

Newsroomtags for crosspoints and router control inputs

When enabling a Switcher Crosspoint or a Router Control device for the template given above and adding some newsroom tags for crosspoints and for router control inputs, the Mosart item created from the template above could have the following fields and transitions:

```
  <fields>
        <field name="input_preview" value="SAT_10" default="SAT_10" fieldtype="LIST"
keylist="routersources" />
            <field name="input_program" value="SAT_10" default="SAT_10"
fieldtype="LIST"
keylist="routersources" />
            <field name="output_preview" value="M_VISU_2" default="M_VISU_2"
fieldtype="LIST"
keylist="routerdestinations" />
            <field name="output_program" value="M_VISU_2" default="M_VISU_2"
fieldtype="LIST"
keylist="routerdestinations" />
            <field name="xpoint" value="CAM2" default="CAM2" fieldtype="LIST"
keylist="mixerinputs" />
      </fields>
     <transitions value="EFFECT" enable="false" rate="9" next_cue_delay="10">
            <transition name="EFFECT">
            <field name="effectname" value="9" fieldtype="LIST" keylist="effects" />
            </transition>
      </transitions>
```

Note: The *keylists* in the Mosart item refers to the keys in the MOS object for a Mosart template.

15.2.3 MOS Object with Clip Templates Merged in a Mosart Template

A single MOS object is created for all Mosart templates of type PACKAGE or VOICEOVER from the default template set. In this example:

- <mosExternalMetadata> node contains the list of variants for the Mosart templates of type PACKAGE and VOICEOVER.
- <objID> of the MOS object is set to *25*, representing a Mosart item type CLIP.
- <objGroup> is the default template set name.
- <type> node has name and category set to “CLIP”, and *templateset* set to the name of the default template set.
- The *variant* for each template is built using some prefixes:
  1) for PACKAGE, the variant is set to MAZ-<template variant>
  2) for VOICEOVER, the variant is set to OFFMAZ-<template variant>

Note: When **GroupByType** is *false*, no MOS object will be sent for templates of type PACKAGE and VOICEOVER.

```
<mosObj>
<objID>LOWERTHIRDS</objID>
<objSlug>LOWERTHIRDS</objSlug>
<mosAbstract />
<objGroup>TEMPLATES</objGroup>
<objType>STILL</objType>
<objTB>50</objTB>
<objRev>1</objRev>
<objDur>0</objDur>
<!-- The status of the MOS object can be NEW, UPDATED or DELETED -->
<status>NEW</status>
<objAir>NOT READY</objAir>
<createdBy/>
<created/>
<changedBy/>
<changed/>
<description />
<mosExternalMetadata>
      <mosScope>PLAYLIST</mosScope>
      <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
      <mosPayload>
         <mosarttemplate>
        <type name="LOWERTHIRDS" category="GRAPHICS">
            <variants fieldtype="LIST" value="AUTOOUT">
            <variant name="AUTOOUT">
                <fields>
                  <field name="graphics_description" fieldtype="TEXT" />
                         <field name="graphics_id" fieldtype="TEXT" />
                        <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00"/>
```

```
                        <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00"/>
                        <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                        <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="KEEPWHILEBACKGROUND">
                <fields>
                  <field name="graphics_description" fieldtype="TEXT" />
                        <field name="graphics_id" fieldtype="TEXT" />
                        <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                        <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                        <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="KEEPWHILESTORY">
                <fields>
                    <field name="graphics_description" fieldtype="TEXT" />
                           <field name="graphics_id" fieldtype="TEXT" />
                         <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                         <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                          <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="NONEAUTOOUT">
                <fields>
                    <field name="graphics_description" fieldtype="TEXT" />
                           <field name="graphics_id" fieldtype="TEXT" />
                         <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                         <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                          <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            </variants>
            <!-- List of active channels (graphics destinations) as configured in
OverlayGraphics Interface-->
```

```
            <keys name="channels">
            <k>WALL_2</k>
            <k>WALL</k>
            <k>TABLE</k>
            <k>DSK</k>
  </keys>
            <keys name="taketypes">
            <k>AUTO</k>
            <k>MANUAL</k>
            </keys>
        </type>
       </mosarttemplate>
    </mosPayload>
</mosExternalMetadata>
</mosObj>
```

The list of channels contains the active destinations (destinations linked to a graphic engine) as configured in the **Overlay Graphics** interface.

For details of configuration, refer to section Show Design and Template Editor .

15.2.4 Structure of the MOS object with Lowerthird Information

Along MOS objects corresponding to the Mosart channel templates, Mosart also sends to NCS a MOS object containing information about lower thirds.

```
<mosObj>
<objID>LOWERTHIRDS</objID>
<objSlug>LOWERTHIRDS</objSlug>
<mosAbstract />
<objGroup>TEMPLATES</objGroup>
<objType>STILL</objType>
<objTB>50</objTB>
<objRev>1</objRev>
<objDur>0</objDur>
<!-- The status of the MOS object can be NEW, UPDATED or DELETED -->
<status>NEW</status>
<objAir>NOT READY</objAir>
<createdBy/>
<created/>
<changedBy/>
<changed/>
<description />
<mosExternalMetadata>
      <mosScope>PLAYLIST</mosScope>
      <mosSchema>http://www.mosartmedialab.no/schema/mositem.dtd</mosSchema>
      <mosPayload>
         <mosarttemplate>
        <type name="LOWERTHIRDS" category="GRAPHICS">
            <variants fieldtype="LIST" value="AUTOOUT">
            <variant name="AUTOOUT">
                <fields>
                  <field name="graphics_description" fieldtype="TEXT" />
                         <field name="graphics_id" fieldtype="TEXT" />
                        <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00"/>
                       <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00"/>
                       <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                       <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="KEEPWHILEBACKGROUND">
                <fields>
                  <field name="graphics_description" fieldtype="TEXT" />
                        <field name="graphics_id" fieldtype="TEXT" />
                       <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                        <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
```

```
                        <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="KEEPWHILESTORY">
                <fields>
                    <field name="graphics_description" fieldtype="TEXT" />
                           <field name="graphics_id" fieldtype="TEXT" />
                         <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                         <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                          <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            <variant name="NONEAUTOOUT">
                <fields>
                    <field name="graphics_description" fieldtype="TEXT" />
                           <field name="graphics_id" fieldtype="TEXT" />
                         <field name="tc_in" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                         <field name="tc_out" fieldtype="TIMECODE" inputmask="mm:ss"
default="00:00" />
                          <field name="channel" value="DSK" fieldtype="LIST"
keylist="channels" />
                         <field name="take" value="AUTO" fieldtype="LIST"
keylist="taketypes" />
                </fields>
            </variant>
            </variants>
            <!-- List of active channels (graphics destinations) as configured in
OverlayGraphics Interface-->
            <keys name="channels">
            <k>WALL_2</k>
            <k>WALL</k>
            <k>TABLE</k>
            <k>DSK</k>
  </keys>
            <keys name="taketypes">
            <k>AUTO</k>
            <k>MANUAL</k>
            </keys>
        </type>
       </mosarttemplate>
    </mosPayload>
</mosExternalMetadata>
</mosObj>
```

Note: The list of channels contains the active destinations (destinations linked to a graphic engine) as configured in **Overlay Graphics** interface.

15.2.5 Octopus

When using template feedback to Octopus, templates are listed with the Mosart Type names only, not the translated names from the NRCS tags list in Newsroomsettings.

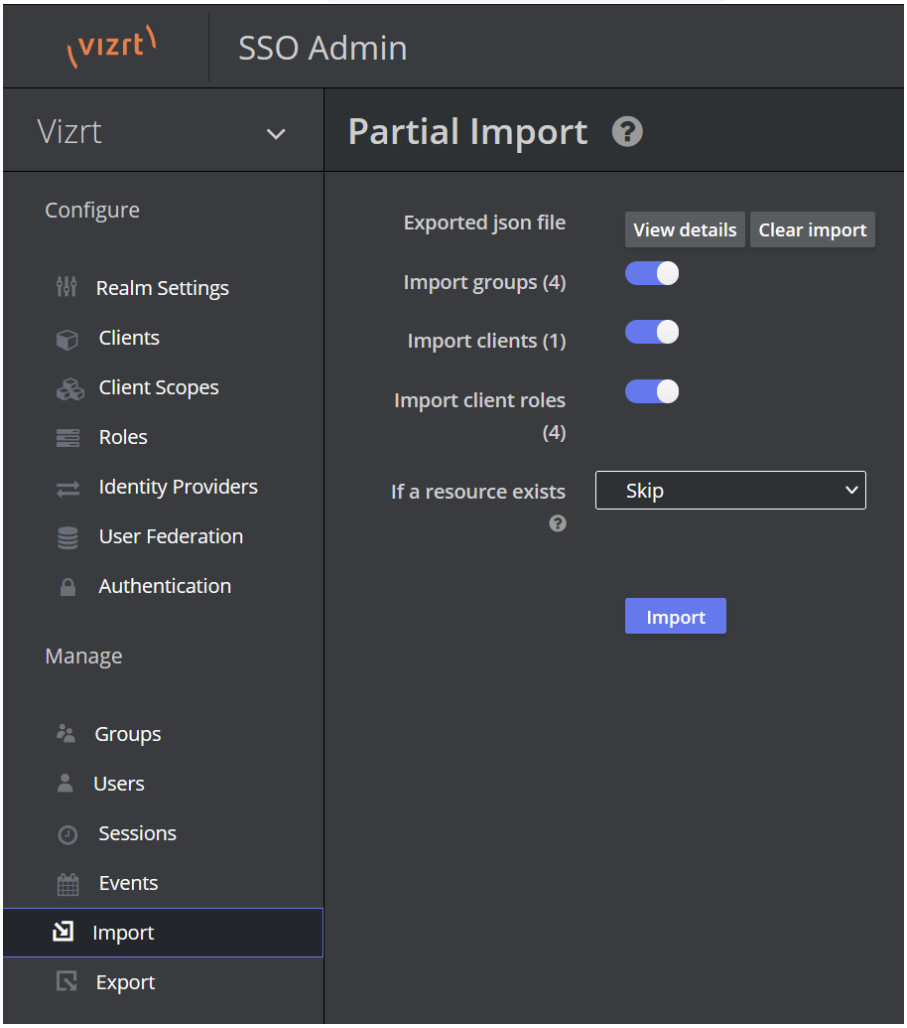
- For Viz Mosart to be able to create the correct timeline objects you must include the Mosart type names as NRCS tag names for all template types.
  As a partial example:

```
<tags>
<tag name="CAM" type="CAMERA" scope="All" />
<tag name="CAMERA" type="CAMERA" scope="All" />
<tag name="OOV" type="VOICEOVER" scope="All" />
<tag name="VOICEOVER" type="VOICEOVER" scope="All" />
```

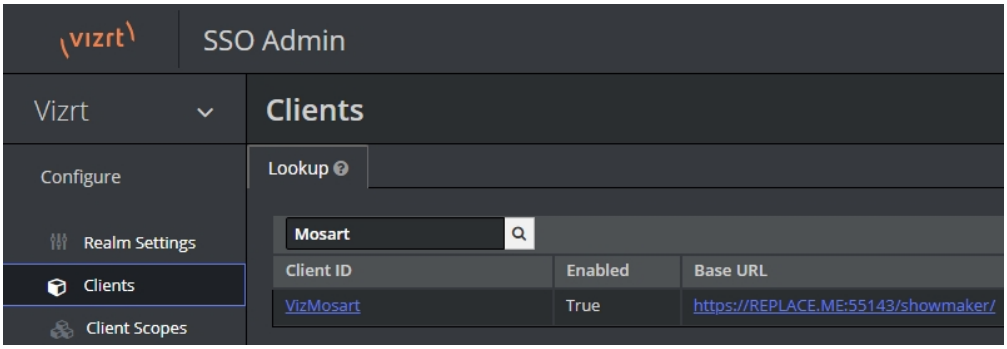
15.3 Vizrt SSO (Keycloak)

To connect the Mosart Web Applications to a Vizrt SSO (for example, when running together with a Viz One authenticated by SSO), perform the following steps:

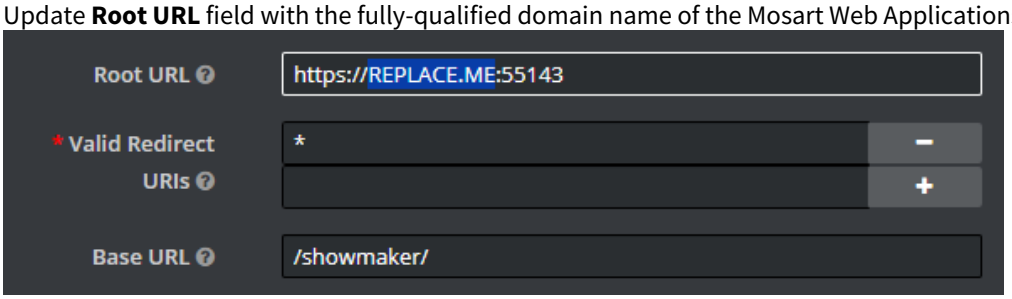
1. Download the `keycloak-mosart-sso.json` file from the same folder that you got the installer on Vizrt's Download server.
2. Log in to the Viz One SSO Admin pages, and go to the **Import** tab in the left menu.
3. Click **Select file** and pick the `keycloak-mosart-sso.json` file.



4. Review the options and when ready, click **Import**.
5. Go to the **Clients** tab, find and open the *VizMosart* client that was just imported.



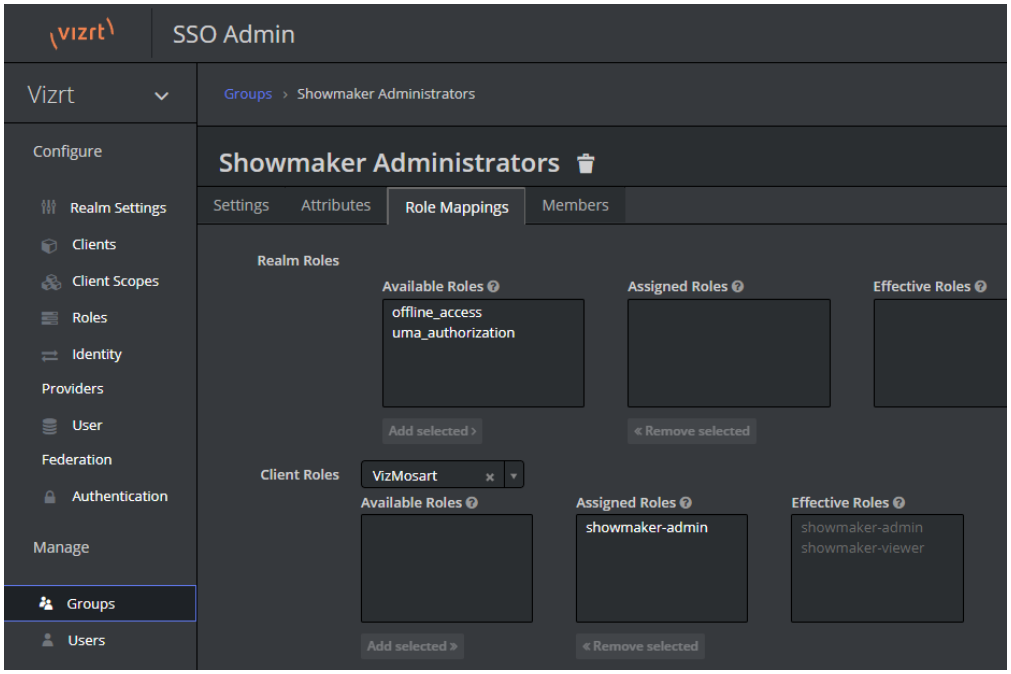
6. Update **Root URL** field with the fully-qualified domain name of the Mosart Web Applications server.



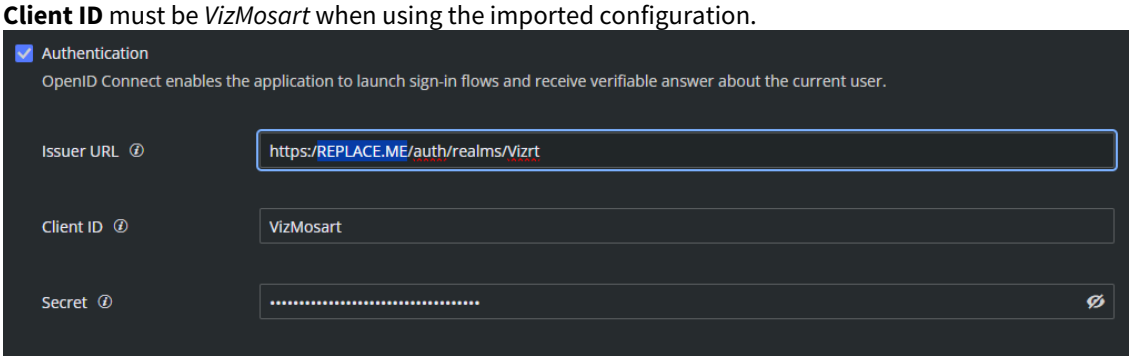
7. Manage users and groups to assign the two Mosart Web Application roles as desired. The roles are defined by the selected **Client ID** *VizMosart*, as shown in the screenshot. The imported configuration includes example groups which you can use as-is, customize, or replace with your own groups.

Info:

- Groups do not directly grant permissions, they assign corresponding *roles* to all group members.
- Roles grant user permissions.
- The *showmaker-admin* role implicitly grants *all* other Pilot roles.

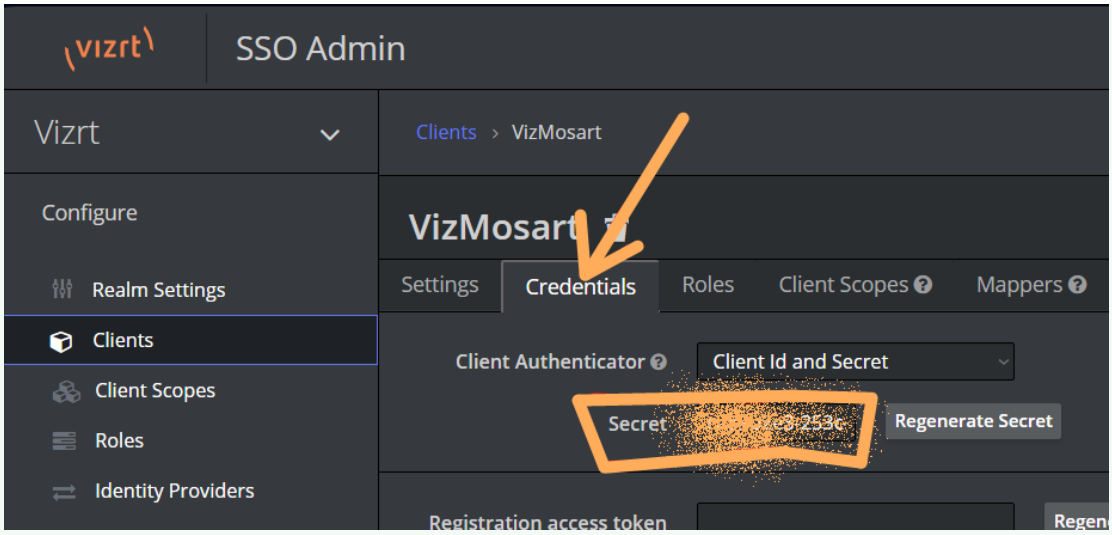


8. Use the Configuration Tool to configure OIDC settings for the target server.
   a. **Issuer URL** is typically the Viz One server's base URL followed by `/auth/realms/Vizrt`.
   b. **Client ID** must be *VizMosart* when using the imported configuration.



Tip: The *Secret* key can be obtained from the Vizrt SSO Administration console.

- Select **Clients > Credentials** (tab).
- Copy the value of **Secret**.


vizrt
SSO Admin
Vizrt
Configure
Realm Settings
Clients
Client Scopes
Roles
Identity Providers
Clients › VizMosart
VizMosart
Settings
Credentials
Roles
Client Scopes
Mappers
Client Authenticator
Client Id and Secret
Secret
Regenerate Secret
Registration access token