Viz Mosart Media Router Guide

Version 5.19



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

2026/11/20

Contents

1 Media Router Introduction 5
1.1 Related Documents 5
1.2 Feedback and Suggestions 5
2 About Media Router 6
2.1 Media Router Naming Convention for Components 7
2.2 Typical Media Router Setup 9
2.3 The Router Concept 10
2.4 Media Router Internal Salvos 13
2.4.1 MMR PendingSalvo (__PENDING) 13
2.4.2 MMR CurrentSalvo (__CURRENT) 13
2.4.3 MMR ProbelSalvo (__PROBEL) 13
2.5 Relation to ProBel Router Protocol 14
3 Media Router Installation and Configuration 15
3.1 Installation 15
3.2 Configuration 15
3.3 Media Router Service Configuration 16
3.3.1 Example Service Configuration 16
3.4 Media Router Database Configuration 24
3.4.1 Media Router Configuration of Viz Mosart Servers: Clients Outports 25
3.4.2 Media Router Configuration of Physical Devices: Source Inports 27
3.4.3 Sample Media Router XML Database 29
4 Viz Mosart Server Media Router Configurations 35
4.1 Viz Mosart Media Router Data Flow 35
4.1.1 AV Automation, local Media Router configuration files 36
4.1.2 AV Automation Media Router Configuration 36
4.1.3 AV Automation, Media Router - Video Server Configuration 38
4.1.4 AV Automation Keyboard Shortcuts 38
4.1.5 OverlayGraphics Media Router Configuration 39
4.1.6 OverlayGraphics Keyboard Shortcuts 40
5 Using Media Router 42
5.1 Running Media Router as a Console Application 43
5.2 Running Media Router as a Windows Service 45
5.3 Media Router Console Commands 46

5.3.1 To activate predefined salvos....46
5.3.2 To set individual crosspoints....46
5.3.3 To get router information....46
5.3.4 Media Router console command examples....47
5.4 Redundancy Setup....48
5.4.1 Related service properties....48
5.5 Mirror Mode....49
5.5.1 Related service properties....49
6 MMT Test Application....50
6.1 To install MMT....50
6.2 MMT Console Commands....51
6.3 MMT Emulating a Viz Mosart Server Client....52
6.3.1 MMT as a second Viz Mosart server client emulator....52
7 Media Router REST Protocol....53
7.1 Viz Mosart Media Router REST Command Format....53
7.2 REST Commands....53
7.2.1 Retrieval Commands....54
7.2.2 Control Commands....57
8 MMR Configuration of Vizrt Graphics....58
8.1 Configuration of Viz MSE....58
8.2 Configuration of Viz Engine....59
8.3 Configuration of Mosart Fullscreen Graphics....60
8.3.1 Mosart Server (as an MMR client) properties....60
8.3.2 Mosart Server Fullscreen (outport) Properties....61
8.4 Configuration of Mosart overlay graphics....61
8.4.1 Mosart Server Overlay (outport) Properties....61
8.5 Viz Graphics Additional Properties....62
8.5.1 Apply Channel Name to Elements....62
8.5.2 Remove Unused Channels Outputs....63

1 Media Router Introduction

Viz Mosart is a unique tool for making repetitive tasks easy. Your production is made effortless for the operator as robotic camera moves, video effects, audio stings, lighting states, and automatically assigned video ports cue for playback to make a perfectly presented production.

This document contains an overview of system prerequisites, installation, and configuration of the **Media Router** within your broadcast environment.

For information about the other components of Viz Mosart, see the *Viz Mosart User Guide* and *Viz Mosart Administrator Guide*.

1.1 Related Documents

- *Viz Mosart Administrator Guide:* Contains information on how to install the **Viz Mosart** software and supported hardware.
- *Viz Mosart User Guide:* Contains information on how to use Viz Mosart in live production.
- *Viz Mosart Functional Specification*

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 Media Router

Media Router allows for device sharing in a limited resource environment. It is used to share broadcast media resources between multiple Viz Mosart servers and other broadcast equipment.

Media Router uses the functionality provided by Viz Mosart server, allowing changes to Viz Mosart-controlled resources to be made dynamically, that is, without restarting any of the Viz Mosart applications.

The equipment types which are controllable by Media Router are:

- Video Servers (VS)
- Overlay Graphics (OG)
- Fullscreen Graphics (FG)
- Robotic Cameras (RC)
- Lights (L)
- Video Wall (W)

Media Router is typically used in the following scenarios:

- To change the resource configuration of a Viz Mosart server installation prior to running a show. That is, to share resources between multiple Viz Mosart server studio installations/galleries.
- In emergencies, to make use of backup resources.

The Media Router resembles a router in that changes are done by setting crosspoints in a router setup, connecting resources (inputs) with Viz Mosart (outputs).

This section contains the following:

- Media Router Naming Convention for Components
- Typical Media Router Setup
- The Router Concept
- Media Router Internal Salvos
- Relation to ProBel Router Protocol

Copyright © 2026 Vizrt

2.1 Media Router Naming Convention for Components

When contacting Vizrt Support, it is helpful to be familiar with Viz Mosart's naming conventions. Some are listed below:

Short Name	Long Name	Description
AVA	AV Automation	Application controlling attached broadcast equipment.
Client	Output client	A client in this context denotes a service given access to one or more sources, using the Media Router. In router terminology a client defines an output of a router matrix.
CurrentSalvo	Current salvo	Internal Media Router state containing validated crosspoints.
GUI	Viz Mosart UI	Main user control application.
Manus Admin	Manus Administrator	Application managing the Viz Mosart rundown.
Media Admin	Media Administrator	Application for monitoring media objects (clips).
OGI	Overlay Graphics Interface	Application that manages an overlay graphics systems.
PendingSalvo	Pending salvo	Internal Media Router state containing all pending crosspoints (corresponding configuration changes to be validated by Mosart servers).
Source	Input source	A dedicated media content provider that will be controlled. Examples of sources in this context are video playout server ports and graphics controllers. In router terminology a source defines an input of a router matrix
MMR	Mosart Media Router	Application that shares broadcast devices between control rooms/ galleries.
MMT	Media Router Test Application	A console application for testing the Media Router service.

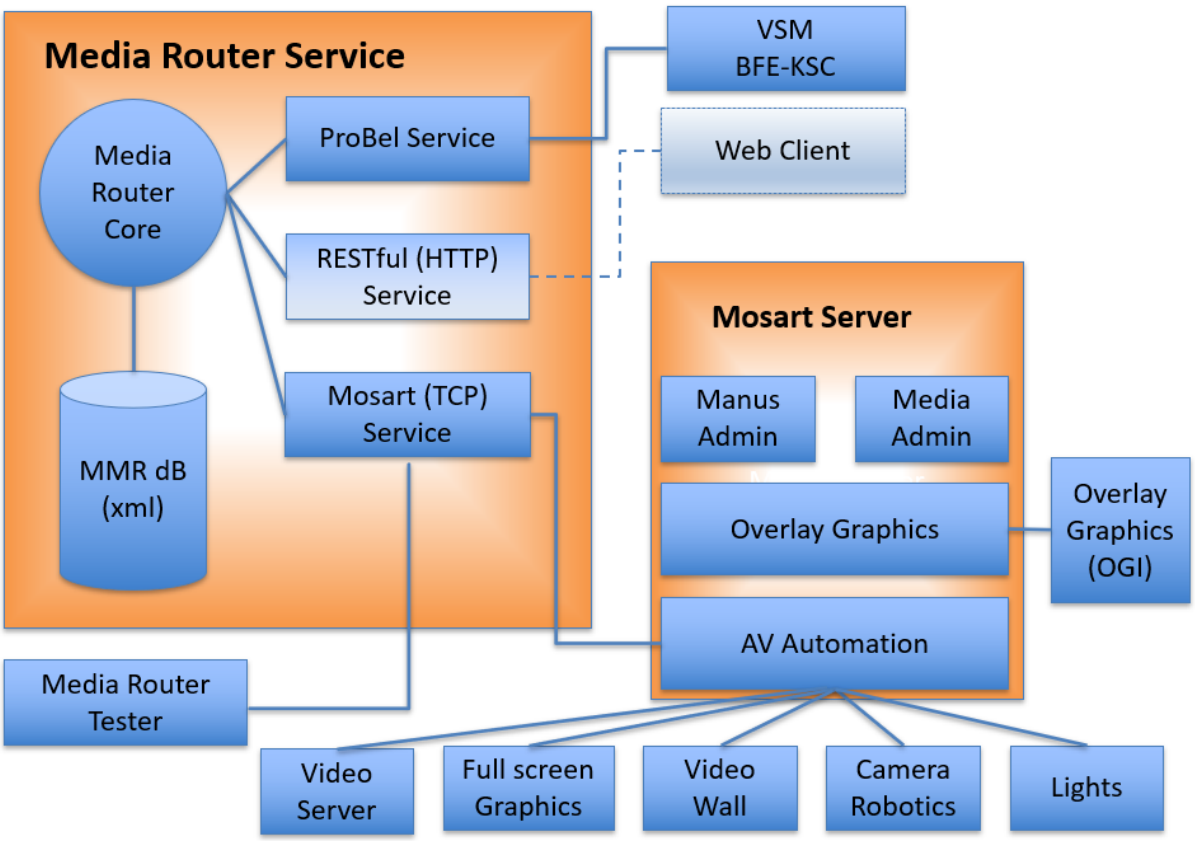
Copyright © 2026 Vizrt

Short Name	Long Name	Description
NCS, NRCS	Newsroom Computer System	A rundown/scheduler application for upstream program (show) planning.
Salvo		A predefined list of crosspoint operations which when executed, all occur simultaneously.

Most of the terms above are explained later in this description.

Copyright © 2026 Vizrt

2.2 Typical Media Router Setup



A typical setup, illustrated above, consists of the following:

- **Media Router service**: This can be configured to support multiple services for connection.
  For most cases, the ProBel SW-P-08 protocol is used for controlling the Media Router whilst the Viz Mosart (TCP) service is used for communicating with Viz Mosart servers.
- **Viz Mosart server**: Using shared resources for device types currently supported by Media Router. The AV Automation application is responsible for communication with Media Router.

Info: AV Automation can forward Media Router configuration to the other Viz Mosart server applications.

- **VSM BFE-KSC**: Virtual studio manager (*external controller*) system for setting and changing Media Router crosspoints, using the ProBel protocol.
- **Media Router Test Application**: A console application element of the Viz Mosart Test Suite, which can be used to verify Media Router configurations.
- **Web client**: Controls the Media Router service using the Media Router REST protocol.

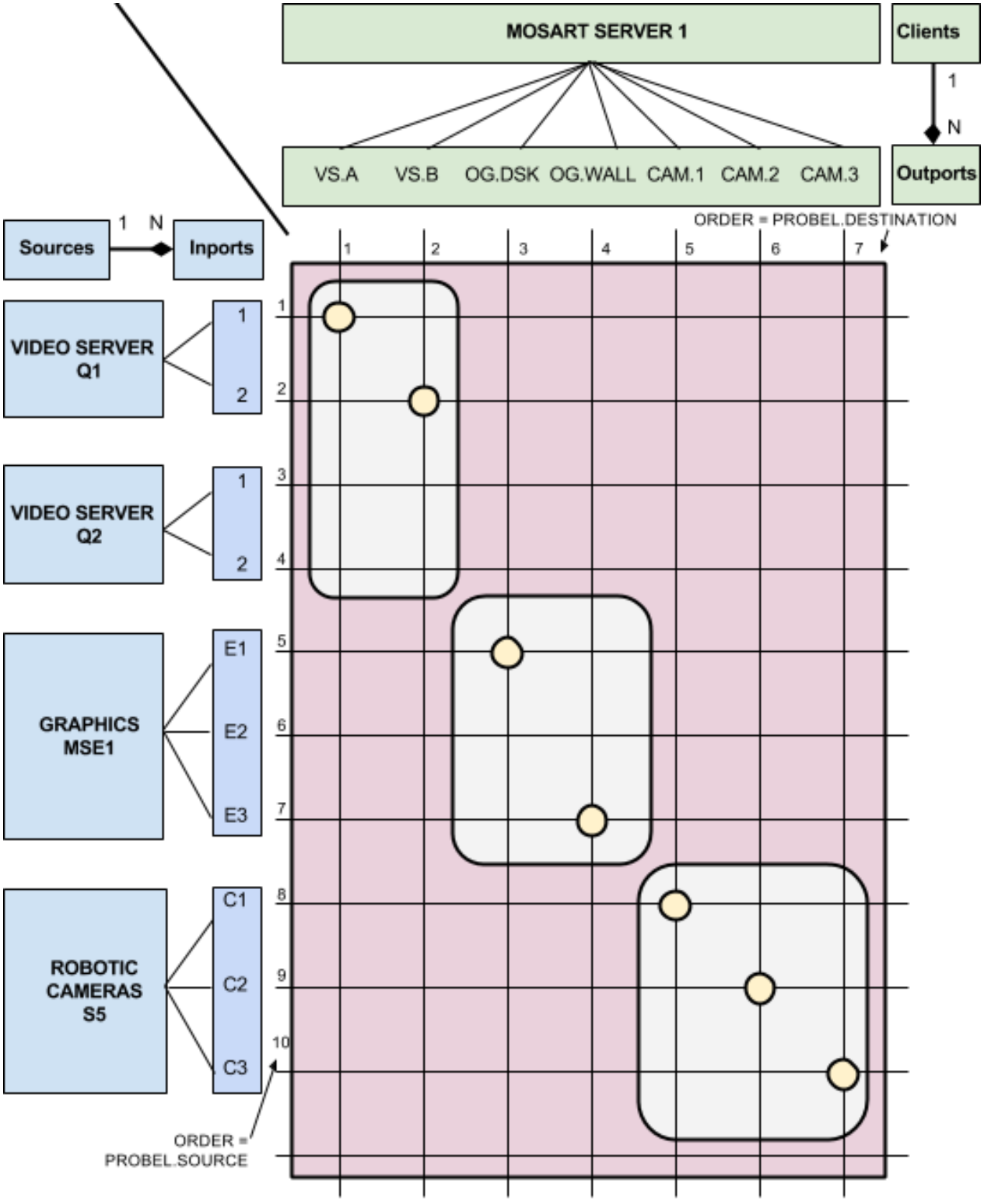
Copyright © 2026 Vizrt

2.3 The Router Concept

Media Router performs dynamic configuration changes to Viz Mosart servers by connecting Viz Mosart device representations with physical devices using a virtual 2D router matrix.

The following figure shows the 2D matrix, where it is only possible to set crosspoints bounded by the grey areas.

Copyright © 2026 Vizrt



Physical devices are localised on the vertical dimension organized as *sources* with *inports*. A *source* may have one or several *inports*. Viz Mosart servers are localized at the horizontal dimension, organized as *clients* with *outports*. A *client* may have one or several *outports*.

Copyright © 2026 Vizrt

In the figure above we have the following configuration:

- *Sources*
  - Two video servers, *Q1* and *Q2* both with two video ports as *inports*
  - One graphics system, *MSE1* with three graphics engines as *inports*
  - One robotic camera system, *S5* with three robotic cameras as *inports.*
- *Clients*
  One single Viz Mosart server (*M1*) with the following device representations as *outports*:
  - Two video ports, A (*VS.A*) and B (*VS.B*)
  - Overlay Graphics for DSK and WALL graphics
  - Three cameras to be controlled by robotic cameras, *CAM1*, *CAM2* and *CAM3*.

By setting crosspoints the following Viz Mosart configuration is accomplished:

- M1.VS.A = VS.Q1.1
- M1.VS.B = VS.Q1.2
- M1.OG.DSK = OG.MSE1.E1
- M1.OG.WALL = OG.MSE1.E3
- M1.RC.CAM1 = RC.S5.C1
- M1.RC.CAM2 = RC.S5.C2
- M1.RC.CAM3 = RC.S5.C3

Copyright © 2026 Vizrt

2.4 Media Router Internal Salvos

The internal state of the Media Router is stored as set of internal salvos (or set of crosspoints).

The following internal Media Router salvos exist:

- MMR PendingSalv(__PENDING) - Represent all current connections.
- MMR CurrentSalv(__CURRENT) - Represent current connections that are verified as valid connections
- MMR ProbelSalvo(__PROBEL) - Represent all current connections set from a controlling ProBel device.

2.4.1 MMR PendingSalvo (__PENDING)

This salvo contains all crosspoints that are currently set in Media Router and shall be treated as the current state of Media Router. Any changes to the PendingSalvo shall lead to configuration changes being sent to the corresponding Viz Mosart servers.

2.4.2 MMR CurrentSalvo (__CURRENT)

This salvo contains all crosspoints currently set and verified by the corresponding Mosart Servers. All crosspoints set in the CurrentSalvo shall represent a valid connection between a Mosart server and a device controlled by the Mosart server. The CurrentSalvo shall always be a subset of the PendingSalvo.

2.4.3 MMR ProbelSalvo (__PROBEL)

This salvo represents all crosspoints set by a controlling ProBel device. This salvo is used for combining multiple crosspoints changes from the controlling ProBel device into a single salvo.

Copyright © 2026 Vizrt

2.5 Relation to ProBel Router Protocol

Media Router is controllable from any service that supports the **ProBel router protocol, SW-P-08**. Relations to the ProBel protocol are performed by assigning crosspoints a unique number with the *Order* attribute.
This order number directly relates to the ProBel protocol as follows:

- Probel.Source = Inport.Order
- Probel.Destination = Outport.Order

Tip: The diagram in section The Router Concept, shows how corresponding order numbers are indicated for all inports and outports.

Copyright © 2026 Vizrt

3 Media Router Installation and Configuration

3.1 Installation

You install a Media Router from the Viz Mosart installer identified as
`VizMosartMediaRouter-<version number>.msi`

3.2 Configuration

MMR Configuration is performed in two xml-based configuration files:

- Media Router service configuration
  in `MediaRouterServiceConfig.xml`
- Media Router Database Configuration
  in `MediaRouterDB.xml`

> **Info**: The *name* of `MediaRouterDB.xml` is configurable within the Media Router service configuration.

Normally both files are placed at:

- `%ProgramData%\Mosart Medialab\ConfigurationFiles`

> **Info**: Older systems use a legacy location `C:\ChannelTemplates`

Copyright © 2026 Vizrt

3.3 Media Router Service Configuration

The service configuration XML file contains the following information:

- **Services:** A list over the protocols that will be active. The following protocols are available:
  - Mosart Media Router Service - Service used to connect to the Viz Mosart service via a proprietary protocol.
  - ProBel Media Router Service - Service allowing Media Router to be controlled via the Probel router protocol, SW-P-08.
  - Rest Media Router Service - Service allowing Media Router to be controlled via a proprietary REST protocol.
- **Database:** The Media Router database specification.
  - Currently supported is a file repository where the Media Router database is stored as a single xml file. Default named *MediaRouterDB.xml* .
- **Service Properties:** A set of properties controlling the behavior of the corresponding services.
- **Properties:** A set of properties controlling the behavior of the Media Router Service (Mosart Media Router Service).

3.3.1 Example Service Configuration

The Media Router service configuration is stored in a single xml file as shown below:

MediaRouterServiceConfig.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<MediaRouterConfig>
  <!-- MMR SERVICES: CONFIGURATION OF ACTIVE PROTOCOLS -->
  <Services>
    <Service type="MosartMediaRouterService" />
    <Service name="vsm" type="ProBelMediaRouterSerialService"
configuration="Type=ProBelMessageServerTransporter;Port=8123" />
    <Service type="RestMediaRouterService" />
  </Services>
  <!-- MMR Database configuration using single xml file -->
  <Database type="File" name="File" configuration="MediaRouterDB.xml" />

  <!--  SERVICE PROPERTIES  -->
  <ServiceProperties name="vsm">     <!-- use name attribute to identify
corresponding service -->
     <item name="Mode" value="Router"/>
     <item name="UpdateCurrentSalvoCrosspoints" value = "true"/>
     <item name="DisableControllerStatusResponse" value = "true"/>
  </ServiceProperties>

  <!-- MosartMediaRouterService PROPERTIES -->
  <Properties>
     <!--  MMR MASTER / SLAVE PROPERTIES -->
```

Copyright © 2026 Vizrt

```
    <item name="Id" value="MR1" />
    <item name="PreferredMaster" value="True" />
    <item name="Slave" value="Name=MR2;Server=localhost;Port=8192" />
    <item name="AutoFailover" value="True" />
    <!--  MOSART SERVER REDUNDANCY PROPERTIES  -->
    <item name="MirrorMode" value="True" />
    <item name="MirrorModeMaster" value="Active" />
    <item name="SynchronizedMirroring" value="False" />
    <!-- GENERAL PROPERTIES -->
    <item name="AutoTake" value="False" />
    <item name="BackupOnStartUp" value="True" />
    <!-- MMR REST PROTOCOL PROPERTIES -->
    <item name="UseAuthorisation" value="False" />
  </Properties>
</MediaRouterConfig>
```

MosartMediaRouterService

One service item of type “MosartMediaRouterService” shall be part of the list of services in order to communicate with Viz Mosart Services. Configuration of the MosartMediaRouterService is done using the Properties section of the MMR service configuration file. The following properties are available:

Property	Description	Default
AutoFailover	Set to true if a slave in a master/slave redundancy setup shall take over control automatically when detecting connectivity problems to the master	true
AutoTake	Set to true if all pending crossspoints automatically should be considered as valid. I.e. a corresponding crosspoints set in the CurrentSalvo	false
BackupOnStartUp	If set a backup of the MMR database will automatically be made when the MMR service is started. Backup files are placed in a directory named MediaRouter. The directory is located in the same directory as the MMR database.	true

Copyright © 2026 Vizrt

Property	Description	Default
Id	Unique identity of the MMR Service instance. Used when two MMR Service instance are used in a master/slave redundancy setup.	GUID
LocalServer	For testing only. Set to true if the MMR service should run locally without any valid network attached.	false
MirrorMode	Set to true to activate mirror mode. If mirror mode is activated then all crosspoints set on a Mosart server in a redundancy setup will also be sent to the other Mosart server. Hence ensuring that the configuration of the two Mosart servers are equal.	false
MirrorModeMaster	In mirror mode, determines when crosspoints set for one Mosart server in a redundancy setup should be reflected to the other Mosart server. Options: • *Active* - only crosspoints set on the currently active Mosart server will be reflected to the other. • *LowestOrder* - only crosspoints set on the Mosart server with lowest order number will be reflected to the other. The order number is set in the MMR database configuration.	Active
PreferredMaster	In a MMR redundancy setup set this to true for the MMR Service instance which should be the main or master service.	false

Copyright © 2026 Vizrt

Property	Description	Default
Slave	Connectionstring to the other MMR instance in a master/slave redundancy setup. Syntax: “Name=<id>”;Server=<hostname>;Port=<port>” where: • *id* - Shall equal the Id property of the other MMR instance • *hostname* - is the hostname of the other MMR instance • *port* - is the tcp/ip listening port of the other MMR instance	empty = no redundancy setup
Synchronized Mirroring	If set and in mirror mode any crosspoint set on a Mosart server will be reflected to the other Mosart server. Regardless whether the Mosart server is active or not.	false
UseAuthorisation	Used to activate authorisation for the MMR REST protocol (4)	false

ProBelMediaRouterService

Used to add control of Media Router via the Probel SW-P-08 protocol. The ProBel service itself is configured using a corresponding ServiceProperties section or via the connection string given by the “configuration” attribute. The following properties are available:

Property	Description	Default
Name	Identifies the service and the corresponding ServiceProperties section. Note that Name is also used to tag log messages related to the service.	

Copyright © 2026 Vizrt

Property	Description	Default
Type	Identifies the how to communicate with the controlling ProBel device. The following options are available: **ProBelMessageClientTransporter** - In this case Media Router connects to the controlling device using a tcp/ip connection. Use Server and Port properties to configure the tcp/ip connection. This is the most common type when connection to a VSM system. **ProBelMessageSerialTransporter** - Same as *ProBelMessageClientTransporter* but uses a serial connection. Use ComPort to identify the COM port. **ProBelMessageServerTransporter** - In this case Media Router listens to connecting devices supporting the ProBel protocol. I.e. Media Router acts like a real router. Use the Port property to specify the listening port. This is common type when connecting to a BFE-KSE system.	
Port	Used by Client and Server Transporter types to identify the tcp/ip port.	10000
ComPort	Used by Serial Transporter type to identify the serial COM port	COM1
Server	Used by Client Transporter type to identify the host of the controlling ProBel device.	localhost

Copyright © 2026 Vizrt

Property	Description	Default

Mode	Identifies how crosspoint changes from the controlling device shall be handled. Only crosspoints that will change the current state of Media Router will lead to any configuration changes.
	The following options are available:
	Router - In this case the Media Router is treated as a physical router:
	• The Media Router salvo “CurrentSalvo” is used for comparison for crosspoint changes.
	• ProBel CrosspointConnected (04) messages are sent to the Probel controlling device to notify any crosspoint changes.
	• The client (like vsm / BFE-KSE) shall query Media Router for its internal state after a successful connection. This shall be done using any of the following SW-P-08 commands:
	•
	  • Crosspoint Interrogate Message (01)
	  • Crosspoint Tally Dump Request Message (21)
	ControlPanel - In this case the Media Router is treated by the ProBel controlling device as “a control panel” used to visualize the current state:
	• The Media Router salvo “PendingSalvo” is used for comparison.
	• ProBel CrosspointConnect (02) messages are sent to the Probel controlling device to notify any crosspoint changes.
	• Media Router shall query the client (like vsm / BFE-KSE) for its initial state after a successful connection. This shall be done using the following SW-P-08 command:
	• Crosspoint Tally Dump Request Message (21)
		Router

UpdateCurrentSalvoCrosspoints	If set will copy any changes to the Media Router “PendingSalvo” to the Media Router “CurrentSalvo”. This is normally used when Mode=”Router” and when it is desirable to treat the pending salvo as the current state.	False
DisableCrosspointChange	If set to true notification of crosspoint changes to the ProBel controlling device is disabled. Otherwise all crosspoint changes are forwarded to the controlling ProBel device according to the value of the Mode property	False
StateSalvo	Specifies the internal Media Router salvo used to track changes for ProBel responses. If set to empty string the internal Media Router salvo will be selected according to the value of the Mode property.	CurrentSalvo

Copyright © 2026 Vizrt

Property	Description	Default
SignalSalvoChangeDelay	Minimum delay between crosspoint messages from the ProBel controlling device for treating the messages to be part of the same salvo.	100 ms
DisableControllerStatusResponse	If set to true will disable any response to the SW-P-08 command: Dual Controller Status Request Message (08). Required when connected to vsm	False
DefaultSourcePort	Port used in Crosspoint Tally Dump Message (22) to denote a not connected destination outport.	-1

RestMediaRouterService

Enables the control of Media Router using Viz Mosart Media Router REST protocol.

Default url for Media Router REST commands is as follows:

```
http://[hostname]:[port]/MosartMediaRouter/Rest.svc/<command>?<params>
```

where:

- **hostname:** Host name where the Media Router service is running
- **port:** TCP/IP port for the Media Router REST service. *Default*: 8094
- **command:** REST command to be processed
- **params:** Optional parameters for the command.

The port may be changed via the `MMediaRouterService.exe.config` configuration file as part of the Media Router service installation.

Command	Description
clients	Retrieves information of all clients.
sources	Retrieves information of all sources.
inports	Retrieves information of all inports or inports connected to a given source
outports	Retrieves information of all outports or outports connected to a given client

Copyright © 2026 Vizrt

Command	Description
salvos	Retrieves information of a single or multiple salvos
current	Retrieves information of the current state of the router
pending	Retrieves information of the pending state of the router
config	Retrieves information of the current or salvo specific configuration for a given client
crosspoints	Retrieves information of changed crosspoints from a given timestamp
setcrosspoint	Sets a specified crosspoint
setsalvo	Fires a specified salvo
status	Returns current Media Router status

For a complete list of REST commands see the Media Router REST Protocol.

Copyright © 2026 Vizrt

3.4 Media Router Database Configuration

Currently the Media Router database configuration is done by editing the Media Router database file, `MediaRouterDb.xml` directly.

The `MediaRouterDb.xml` file is divided into distinct categories for **Clients > Outports and Sources > Inports**. An *outport* is linked to its *client* and an *inport* is linked to its *source* using the *Id* property.

The tables in the following subsections specify a sample setup with devices to be controlled along with two Viz Mosart servers in a redundancy setup (main/backup).

It is recommended to create such tables before doing the Media Router configuration.

Info: The following abbreviations are used to make it easier to identify connection type or category:

- VS = Video Server
- OG = Overlay Graphics
- FG = Fullscreen Graphics
- RC = Robotic Cameras
- L = Lights
- W = Video Wall

Copyright © 2026 Vizrt

3.4.1 Media Router Configuration of Viz Mosart Servers: Clients Outports

The tables below shows a sample setup of two Viz Mosart servers in a main/backup redundancy setup. Viz Mosart servers shall use two video ports, two cameras controllable via camera robotics, overlay graphics with two destinations (DSK and WALL) and one full screen graphics destination (FULL1).

Both Viz Mosart servers shall have identical setup and make use of the Media Router Mirror mode functionality. Hence it will only be necessary to specify outports for one of the servers. See also Corresponding XML Content.

Viz Mosart Servers (Client specifications)

Gallery	Mosart Server	Client.Id	Order	ConnectionString
M1				
- Server	M1.A	M1.A	1	Server=10.116.5.10;Port=8099
- Server	M1.B	M1.A	2	Server=10.116.5.11;Port=8099

Note: The Port number in the above **ConnectionString** values must also be configured on the Mosart Server, as the *Port number* for *Media Router* in AV Automation's **Device Properties > General tab**.

- See the section AV Automation > AV Automation Device Properties > General in the *Viz Mosart Administrator Guide*.
- Further details are also provided in this MMR Guide in section AV Automation Media Router Configuration.

Viz Mosart Server, Device Handlers (Outport specifications)

Mosart Server	Order	Client.Id	Outport.Id	Device Name	Description
M1.A					
- VideoPort	1	M1.A	M1.A.VA	A	Mosart video port A
- VideoPort	2	M1.A	M1.A.VB	B	Mosart video port B
- Camera	3	M1.A	M1.A.C1	1	Camera 1 using camera robotics
- Camera	4	M1.A	M1.A.C2	2	Camera 2 using camera robotics

Copyright © 2026 Vizrt

Mosart Server	Order	Client.Id	Outport.Id	Device Name	Description
- Overlay GFX	5	M1.A	M1.A.DSK	1	Overlay graphics with destination = DSK
- Overlay GFX	6	M1.A	M1.A.WALL	2	Overlay graphics with destination = WALL
- Fullsc GFX	7	M1.A	M1.A.FULL1	1	Fullscreen graphics, FULL1

Note that a mapping between the DeviceName property and graphics destination is required for OverlayGraphics. See the MMR section of Overlay Graphics Configuration.

Copyright © 2026 Vizrt

3.4.2 Media Router Configuration of Physical Devices: Source Inports

The tables below shows a sample setup of two video servers, two robotic camera controllers, one graphics system dedicated for overlay graphics and one graphics system dedicated for fullscreen graphics. See also corresponding xml content.

Video Servers: category=Video

Category	Order	Source.Id	Inport.Id	Type	DeviceName	ConnectionString
Server		VS.Q1		Quantel		SerialNo=20912;IOR= …
- Port	1	VS.Q1	VS.Q1.1	Quantel	1	
- Port	2	VS.Q1	VS.Q1.2	Quantel	2	
Server		VS.Q2		Quantel		SerialNo=20914;IOR= …
- Port	3	VS.Q2	VS.Q2.1	Quantel	1	
- Port	4	VS.Q2	VS.Q2.2	Quantel	2	

Robotic Cameras: category=RoboticCameras

Category	Order	Source.Id	Inport.Id	Type	DeviceName	ConnectionString
Controller		RC.S5		FX-Motion		
- Camera	5	RC.S5	RC.S5.C1	FX-Motion	1	Host=10.116.5.111
- Camera	6	RC.S5	RC.S5.C2	FX-Motion	2	Host=10.116.5.112
Controller		RC.S6		FX-Motion		
- Camera	7	RC.S6	RC.S6.C1	FX-Motion	1	Host=10.116.5.121

Copyright © 2026 Vizrt

Category	Order	Source.Id	Inport.Id	Type	DeviceName	ConnectionString
- Camera	8	RC.S6	RC.S6.C2	FX-Motion	2	Host=10.116.5.122

Overlay Graphics: category=OverlayGraphics

Category	Order	Source.Id	Inport.Id	Type	DeviceName	ConnectionString
Controller		OG.MSE1		Vizrt		
- Engine	9	OG.MSE1	OG.MSE1.E1	Vizrt	1	Host=10.116.5.141;Port=6100
- Engine	10	OG.MSE1	OG.MSE1.E2	Vizrt	2	Host=10.116.5.142;Port=6100

Fullscreen Graphics: category=FullscreenGraphics

Category	Order	Source.Id	Inport.Id	Type	DeviceName	ConnectionString
Controller		FG.MSE2		Vizrt		
- Engine	11	FG.MSE2	FG.MSE2.E1	Vizrt	1	Host=10.116.5.151;Port=6100
- Engine	12	FG.MSE2	FG.MSE2.E2	Vizrt	2	Host=10.116.5.152;Port=6100

The router configuration contains necessary connection strings for inputs and crosspoint properties used by the Viz Mosart server. These connection strings are identical to the ones used in the Viz Mosart static configuration.
For further details refer to the *Viz Mosart Administrator Guide*, section AV Automation > Device Connection Strings.

Copyright © 2026 Vizrt

3.4.3 Sample Media Router XML Database

MediaRouterDb.xml

```
<MediaRouterDBConfig>
    <Sources>
        <!--VIDEO SERVERS -->
        <Value>
            <item id = "VS.Q1" name = "VS.Q1" type = "Quantel">

<ConnectionString>Name=Q1;Type=Quantel;SerialNo=20912;Mode=Player;IOR=http://
quantel:@10.211.112.112/ZoneManager.ior;Config=ClipServerQuantel.xml</
ConnectionString>
            </item>
        </Value>
        <Value>
            <item id = "VS.Q2" name = "VS.Q2" type = "Quantel">

<ConnectionString>Name=Q2;Type=Quantel;SerialNo=20914;Mode=Player;IOR=http://
quantel:@10.211.112.114/ZoneManager.ior;Config=ClipServerQuantel.xml</
ConnectionString>
            </item>
        </Value>
        <!--ROBOTIC CAMERA CONTROLLERS -->
        <!--Studio 5: FxMotion robotic camera controller -->
        <Value>
            <item id = "RC.S5" name = "RC.S5" type = "FX-Motion" category =
"RoboticCameras"/>
        </Value>
        <!--Studio 6: FxMotion robotic camera controller -->
        <Value>
            <item id = "RC.S6" name = "RC.S6" type = "FX-Motion" category =
"RoboticCameras"/>
        </Value>
        <!--OVERLAY GRAPHICS ENGINES -->
        <!--Vizrt MSE Controller localhost:8594 for checking profile use http://
localhost:8580/app/vdomconfig/vdomconfig.html -->
        <Value>
            <item id = "OG.MSE1" name = "OG.MSE1" type = "VIZRT" category =
"OverlayGraphics">

<ConnectionString>Server=localhost;Port=8594;Playlist=Mosart_Overlay</
ConnectionString>
            </item>
        </Value>
        <!--FULLSCREEN GRAPHICS ENGINES -->
        <Value>
            <item id = "FG.MSE2" name = "FG.MSE2" type = "Vizrt" category =
"FullscreenGraphics">
```

Copyright © 2026 Vizrt

```
                <ConnectionString>Host=10.211.112.130;Port=6100</ConnectionString>
            </item>
        </Value>
        <!--VIDEO WALL -->
        <Value>
            <item id = "VW.PA" name = "VW.PA" type = "Pandora" category =
"VideoWall">

<ConnectionString>Server=10.211.112.140:6162;Backup=10.211.112.150:6162</
ConnectionString>
            </item>
        </Value>
    </Sources>
    <Clients>
        <!--MOSART SERVER, GALLERY 1: MAIN (A) AND BACKUP (B) -->
        <Value>
            <item id = "M1.A" name = "M1.A" order = "1" type = "Mosart">
                <ConnectionString>Server=10.116.5.10;Port=8099</ConnectionString>
                <Slave>M1.B</Slave>
            </item>
        </Value>
        <Value>
            <item id = "M1.B" name = "M1.B" order = "2" type = "Mosart">
                <ConnectionString>Server=10.116.5.20;Port=8099</ConnectionString>
                <Slave>M1.A</Slave>
            </item>
        </Value>
    </Clients>
    <Inports>
        <!--VIDEO PORTS -->
        <!--Quantel, Q1: Video ports -->
        <Value>
            <item id = "VS.Q1.1" name = "VS.Q1.1" order = "1" device = "VS.Q1"
deviceName = "1" crosspoint = "HD1"/>
        </Value>
        <Value>
            <item id = "VS.Q1.2" name = "VS.Q1.2" order = "2" device = "VS.Q1"
deviceName = "2" crosspoint = "HD2"/>
        </Value>
        <!--Quantel, Q2: Video ports -->
        <Value>
            <item id = "VS.Q2.1" name = "VS.Q2.1" order = "3" device = "VS.Q2"
deviceName = "1" crosspoint = "HD1"/>
        </Value>
        <Value>
            <item id = "VS.Q2.2" name = "VS.Q2.2" order = "4" device = "VS.Q2"
deviceName = "2" crosspoint = "HD2"/>
        </Value>
        <!--ROBOTIC CAMERAS -->
        <!--Studio 5: FxMotion robotic cameras -->
        <Value>
            <item id = "RC.S5.1" name = "RC.S5.1" order = "5" device = "RC.S5"
deviceName = "1" category = "RoboticCameras">
```

Copyright © 2026 Vizrt

```
                <ConnectionString>Host=10.116.5.111</ConnectionString>
            </item>
        </Value>
        <Value>
            <item id = "RC.S5.2" name = "RC.S5.2" order = "6" device = "RC.S5"
deviceName = "2" category = "RoboticCameras">
                <ConnectionString>Host=10.116.5.112</ConnectionString>
            </item>
        </Value>
        <!--Studio 6: FxMotion robotic cameras -->
        <Value>
            <item id = "RC.S6.1" name = "RC.S6.1" order = "7" device = "RC.S6"
deviceName = "1" category = "RoboticCameras">
                <ConnectionString>Host=10.116.5.121</ConnectionString>
            </item>
        </Value>
        <Value>
            <item id = "RC.S6.2" name = "RC.S6.2" order = "8" device = "RC.S6"
deviceName = "2" category = "RoboticCameras">
                <ConnectionString>Host=10.116.5.122</ConnectionString>
            </item>
        </Value>
        <!--OVERLAY GRAPHICS ENGINES: Vizrt, 1xMSE + 2 Viz Engines (DSK,WALL) -->
        <Value>
            <item id = "OG.MSE1.E1" name = "OG.MSE1.E1" order = "9" device =
"OG.MSE1" deviceName = "1" category = "OverlayGraphics">
                <ConnectionString>Host=10.116.5.141;Port=6100</ConnectionString>
            </item>
        </Value>
        <Value>
            <item id = "OG.MSE1.E2" name = "OG.MSE1.E2" order = "10" device =
"OG.MSE1" deviceName = "2" category = "OverlayGraphics">
                <ConnectionString>Host=10.116.5.142;Port=6100</ConnectionString>
            </item>
        </Value>
        <!--FULLSCREEN GRAPHICS ENGINES -->
        <Value>
            <item id = "FG.MSE2.E1" name = "FG.MSE2.E1" order = "11" device =
"FG.MSE2" deviceName = "1" category = "FullscreenGraphics">
                <ConnectionString>Host=10.116.5.151;Port=6100;Encoding=UTF-8</
ConnectionString>
            </item>
        </Value>
        <Value>
            <item id = "FG.MSE2.E2" name = "FG.MSE2.E2" order = "12" device =
"FG.MSE2" deviceName = "2" category = "FullscreenGraphics">
                <ConnectionString>Host=10.116.5.152;Port=6100;Encoding=UTF-8</
ConnectionString>
            </item>
        </Value>
        <!--VIDEO WALL -->
        <Value>
```

Copyright © 2026 Vizrt

```
            <item id = "VW.PA.1" name = "VW.PA.1" order = "13" device = "VW.PA"
deviceName = "1" category = "VideoWall"/>
        </Value>
    </Inports>
    <Outports>
        <!--GALLERY 1: MOSART MAIN SERVER -->
        <!--GALLERY 1: MOSART VIDEO PORTS -->
        <Value>
            <item id = "M1.A.VA" name = "M1.A.VA" order = "1" device = "M1.A"
deviceName = "A" category = "Video"/>
        </Value>
        <Value>
            <item id = "M1.A.VB" name = "M1.A.VB" order = "2" device = "M1.A"
deviceName = "B" category = "Video"/>
        </Value>
        <!--GALLERY 1: ROBOTIC CAMERAS -->
        <Value>
            <item id = "M1.A.C1" name = "M1.A.C1" order = "5" device = "M1.A"
deviceName = "1" category = "RoboticCameras"/>
        </Value>
        <Value>
            <item id = "M1.A.C2" name = "M1.A.C2" order = "6" device = "M1.A"
deviceName = "2" category = "RoboticCameras"/>
        </Value>
        <!--GALLERY 1: OVERLAY GRAPHICS ENGINES -->
        <Value>
            <item id = "M1.A.DSK" name = "M1.A.DSK" order = "7" device = "M1.A"
deviceName = "1" category = "OverlayGraphics"/>
        </Value>
        <Value>
            <item id = "M1.A.WALL" name = "M1.A.WALL" order = "8" device = "M1.A"
deviceName = "2" category = "OverlayGraphics"/>
        </Value>
        <!--GALLERY 1: FULLSCREEN GRAPHICS ENGINES -->
        <Value>
        <item id="M1.A.FULL1" name="M1.A.FULL1" order="9" device="M1.A"
deviceName="1" category="FullscreenGraphics" />
        </Value>
    </Outports>
    <Salvos>
        <!-- VIDEO SERVER TEST SALVO: Sets Quantel ports, Swaps ports from SALVO
VS.BA  -->
        <Value>
            <item id = "VS.AB" name = "VS.AB" order = "0" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "VS.Q2.1" outport = "M1.A.VA"/>
                    </Value>
                    <Value>
                        <item inport = "VS.Q2.2" outport = "M1.A.VB"/>
                    </Value>
                </Crosspoints>
            </item>
```

```
        </Value>
        <!--VIDEO SERVER TEST SALVO: Sets Quantel ports, Swaps ports from SALVO VS.AB
-->
        <Value>
            <item id = "VS.BA" name = "VS.BA" order = "1" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "VS.Q2.2" outport = "M1.A.VA"/>
                    </Value>
                    <Value>
                        <item inport = "VS.Q2.1" outport = "M1.A.VB"/>
                    </Value>
                </Crosspoints>
            </item>
        </Value>
        <!--OVERLAYGRAPHICS TEST SALVO: Sets DSK,WALL engines, Swaps engines from
SALVO OG.W21 -->
        <Value>
            <item id = "OG.W12" name = "OG.W12" order = "2" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "OG.MSE1.E1" outport = "M1.A.DSK"/>
                    </Value>
                    <Value>
                        <item inport = "OG.MSE1.E2" outport = "M1.A.WALL"/>
                    </Value>
                </Crosspoints>
            </item>
        </Value>
        <!--OVERLAYGRAPHICS TEST SALVO: Sets DSK,WALL engines, Swaps engines from
SALVO OG.W12 -->
        <Value>
            <item id = "OG.W21" name = "OG.W21" order = "3" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "OG.MSE1.E2" outport = "M1.A.DSK"/>
                    </Value>
                    <Value>
                        <item inport = "OG.MSE1.E1" outport = "M1.A.WALL"/>
                    </Value>
                </Crosspoints>
            </item>
        </Value>
        <!--ROBOTIC CAMERA TEST SALVO: Swaps Cameras from SALVO RC.C21 -->
        <Value>
            <item id = "RC.C12" name = "RC.C12" order = "4" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "RC.S5.1" outport = "M1.A.C1"/>
                    </Value>
                    <Value>
                        <item inport = "RC.S5.2" outport = "M1.A.C2"/>
                    </Value>
```

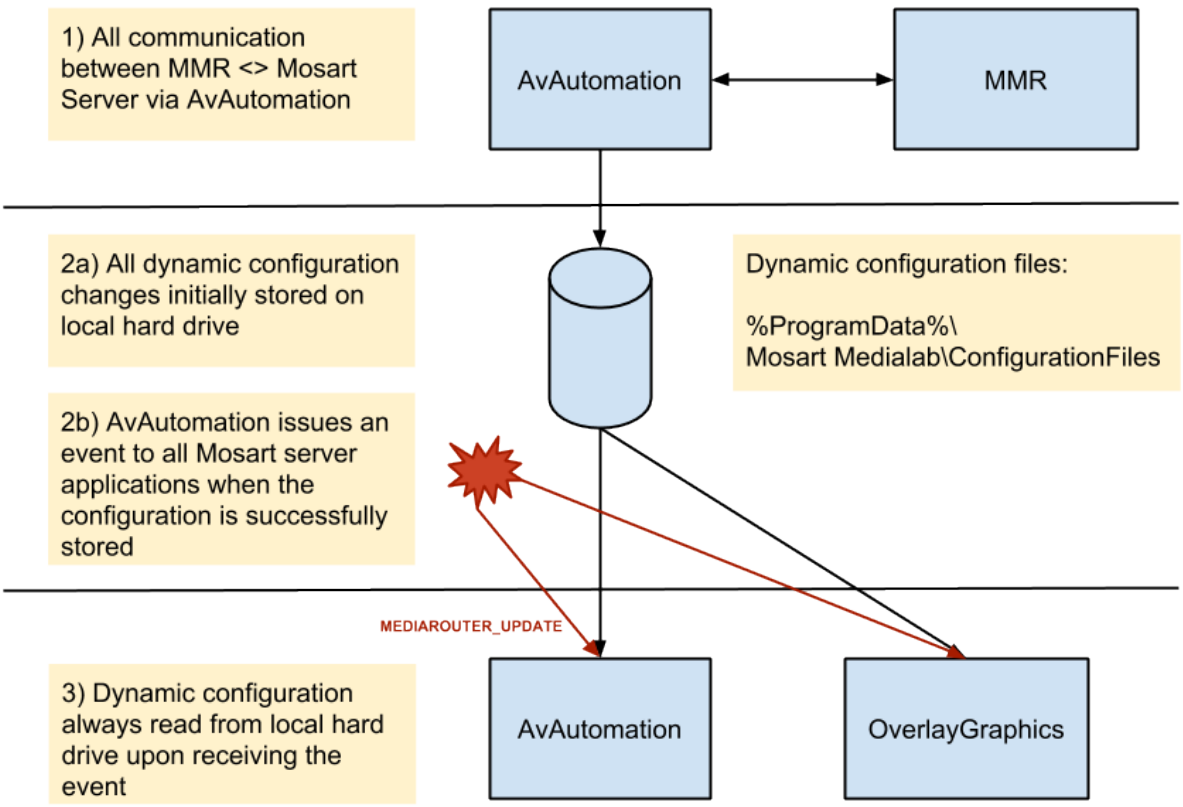
Copyright © 2026 Vizrt

```
                </Crosspoints>
            </item>
        </Value>
        <!--ROBOTIC CAMERA TEST SALVO: Swaps Cameras from SALVO RC.C12 -->
        <Value>
            <item id = "RC.C21" name = "RC.C21" order = "5" ignore = "false">
                <Crosspoints>
                    <Value>
                        <item inport = "RC.S5.2" outport = "M1.A.C1"/>
                    </Value>
                    <Value>
                        <item inport = "RC.S5.1" outport = "M1.A.C2"/>
                    </Value>
                </Crosspoints>
            </item>
        </Value>
    </Salvos>
</MediaRouterDBConfig>
```

4 Viz Mosart Server Media Router Configurations

4.1 Viz Mosart Media Router Data Flow

The illustration presents data flow between Media Router and a Viz Mosart Server.



The Media Router <> Viz Mosart data flow has the following characteristics:

1. All communication between Media Router and Viz Mosart Server is done via the AV Automation application. Hence the Media Router configuration of a Viz Mosart Server is located in AV Automation.
2. When AV Automation receives dynamical configuration changes from Media Router two distinct steps are carried out:
3.
    a. The received dynamic configuration is stored to the local hard drive
    b. A MEDIAROUTER_UPDATE event is issued to all Viz Mosart applications.
4. When an MEDIAROUTER_UPDATE event is received by a Viz Mosart application any corresponding dynamic configuration is obtained from the local hard drive.

Advantages of this approach are as follows:

- If connection to Media Router is lost or missing, the Viz Mosart server will use the last stored configuration.

Copyright © 2026 Vizrt

- Changes to the stored dynamic configuration can be used for testing and verifying connections to devices to be controlled by the Viz Mosart server.

The remainder of this section presents the following Viz Mosart Server Media Router topics:

- AV Automation, local Media Router configuration files
- AV Automation Media Router Configuration
- AV Automation, Media Router - Video Server Configuration
- AV Automation Keyboard Shortcuts
- OverlayGraphics Media Router Configuration
- OverlayGraphics Keyboard Shortcuts

4.1.1 AV Automation, local Media Router configuration files

Whenever a Media Router configuration change is received from the Media Router, the corresponding configuration is stored in one or several files on the local hard drive. This is indicated in point **2a** in the figure above.
These configuration files are:

- `MediaRouterConfigVideo.xml` - contains the current video server configuration.
- `MediaRouterConfigOverlayGraphics.xml` - contains the current overlay graphics configuration.
- `MediaRouterConfigFullscreenGraphics.xml` - contains the current fullscreen graphics configuration.
- `MediaRouterConfigRoboticCameras.xml` - contains the current camera robotics configuration.
- `MediaRouterConfigVideoWall.xml` - contains the current video wall configuration.
- `MediaRouterConfigLights.xml` - contains the current lights configuration.
- `MediaRouterConfigSubtitles.xml` - contains the current subtitles configuration.

All these configuration files shall be located in the `%ProgramData%\Mosart Medialab\ConfigurationFiles` directory.

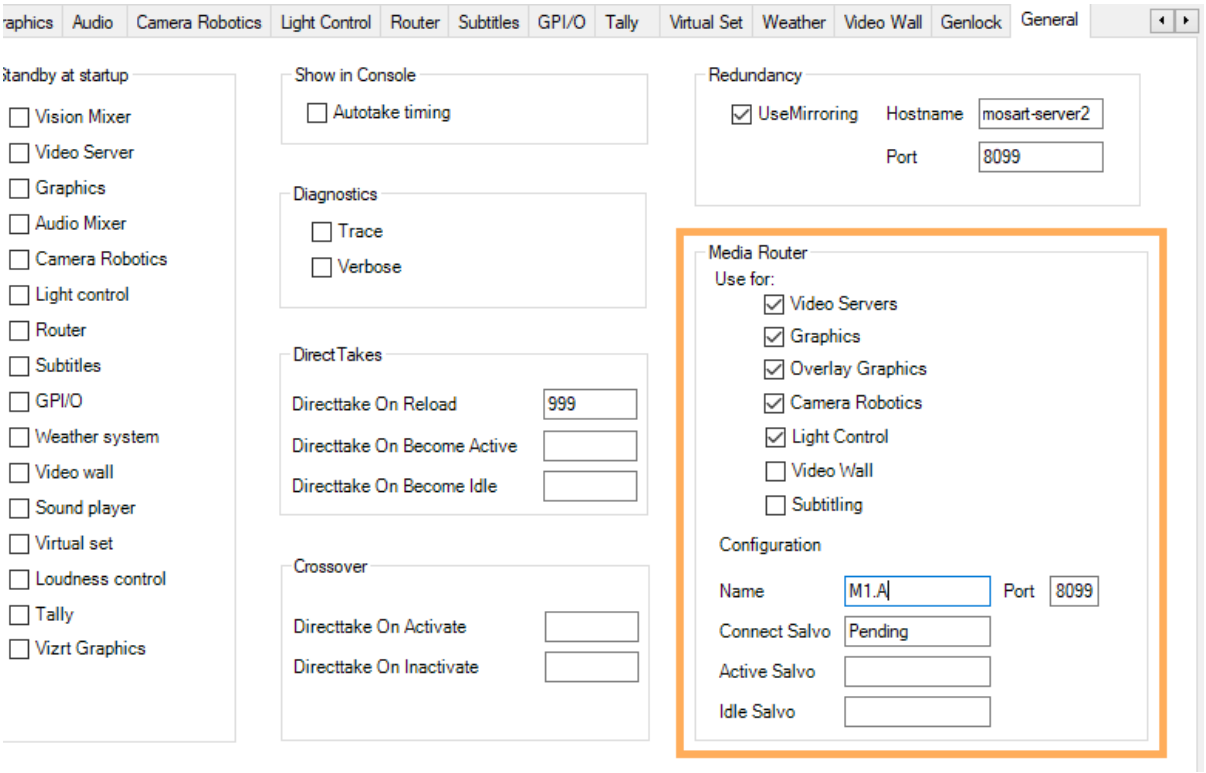
4.1.2 AV Automation Media Router Configuration

To configure devices for Media Router

All Media Router configuration is performed in AV Automation.

1. From the AV Automation console menu bar, navigate to **Device > Preferences > General** tab.
   The red rectangle area (screenshot below) is for Media Router configuration.

Copyright © 2026 Vizrt



2. Select the corresponding check-box for each of the supporting device types where Media Router is to be enabled.
   (In the example above Media Router is activated for *video servers*, *fullscreen graphics*, *overlay graphics*, *camera robotics* and *lights*.

> **Note**: Enabling Media Router for overlay graphics also requires configuration in the Overlay Graphics (Viz Mosart OGI) application

Other Media Router configuration properties are as follows:

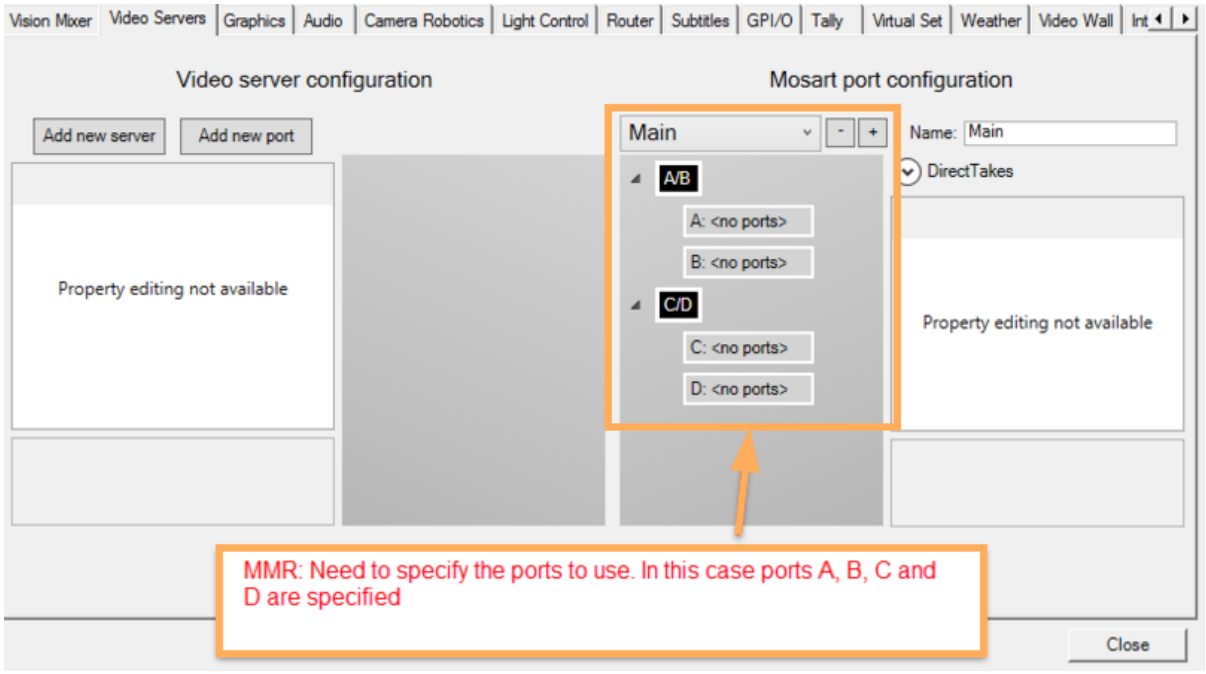
Property	Description
Name	The name of the Viz Mosart server. This name is used in the Media Router database configuration to identify the Viz Mosart server. This must match the corresponding Mosart server configuration in the Media Router database (MediaRouterDb.xml)
Port	The TCP/IP listening port used by AV Automation to listen to any Media Router connections. This property is used by the Media Router database configuration.
Connect Salvo	Salvo requested by Viz Mosart server after a successful connection to Media Router. Default: *Pending*
Active Salvo	Salvo requested by Viz Mosart server when activated. Default: *<empty>*

Copyright © 2026 Vizrt

Property	Description
Idle Salvo	Salvo requested by Viz Mosart server when become idle. Default: *<empty>*

4.1.3 AV Automation, Media Router - Video Server Configuration

When Media Router is used for video servers it is necessary to specify the ports that are to be used by the Viz Mosart Server. This is done in the **Video Servers** configuration found in **Device > Properties** dialog:



Note: In the example above, it is only necessary to specify the Mosart port configuration (right side). The static video server configuration (left side) may be left empty.

4.1.4 AV Automation Keyboard Shortcuts

The following keyboard shortcuts in AV Automation can be used to verify and test a Media Router setup:

Shortcut	Description
CTRL+SHIFT+V	Reloads video server configuration
CTRL+SHIFT+G	Reloads fullscreen graphics configuration

Copyright © 2026 Vizrt

Shortcut	Description
CTRL+SHIFT+R	Reloads camera robotics configuration
CTRL+SHIFT+L	Reloads lights configuration
CTRL+SHIFT+W	Reloads video wall configuration
CTRL+SHIFT+T	Reloads subtitles configuration
CTRL+SHIFT+D	Stores current configuration to local hard drive. I.e. emulate configuration received from Media Router.

Note: The shortcuts also work when AV Automation uses the static configuration.

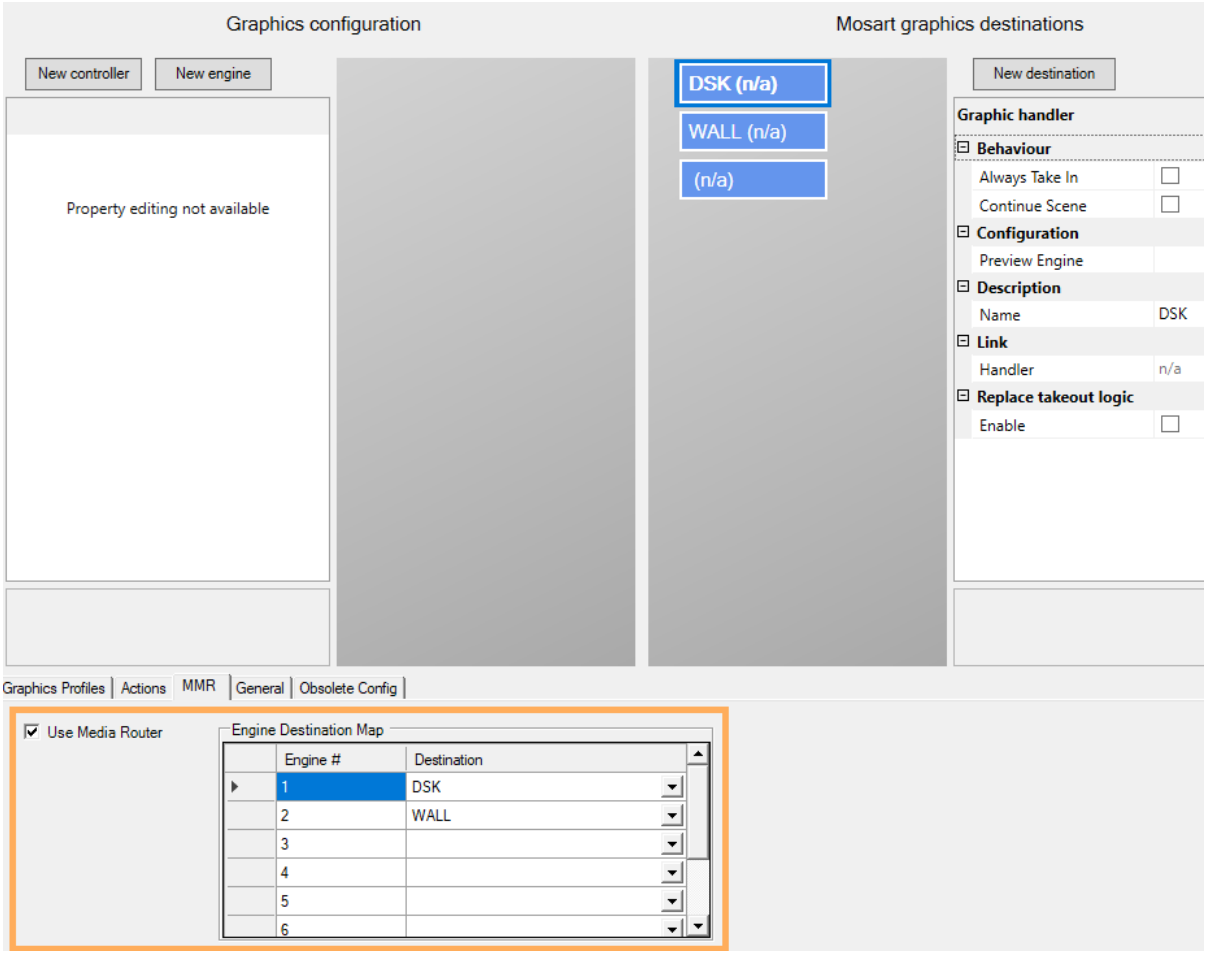
4.1.5 OverlayGraphics Media Router Configuration

To enable dynamic / Media Router configuration in OverlayGraphics the following has to be done:

- Media Router for OverlayGraphics need to be enabled in AV Automation. See AV Automation, Media Router - Video Server Configuration above.
- In OverlayGraphics Settings:
- Enable Media Router by checking the **Use Media Router** check box.
    - Optional: Configure the engine destination map with required destinations (right side). This is only necessary if engine numbers are used in the MMR Mosart overlay graphics configuration (Outport.DeviceName)

The following figure shows a configuration where Engine 1 and 2 is used for DSK and WALL graphics destinations respectively.

Copyright © 2026 Vizrt



Note: When using Media Router for OverlayGraphics, it is not necessary to create any Graphics configuration (left side).
However it is required to specify all graphics destinations (right side) together with an optional mapping between engine number and graphics destination.

The DeviceName property as part of the MMR Outport configuration is used to either specify the graphics destination directly (DSK,WALL) or indirectly via engine numbers (1,2).

Note: Enable Media Router for OverlayGraphics also requires OverlayGraphics to be enabled in AV Automation.

4.1.6 OverlayGraphics Keyboard Shortcuts

Shortcut	Description
CTRL+SHIFT+G	Reloads overlay graphics configuration and performs a reconnect.

Copyright © 2026 Vizrt

Shortcut	Description
CTRL+SHIFT+Add	Dumps the current configuration to file used to store the MMR configuration. May be used to verify MMR without presence of MMR.

Note: This shortcut also works when OverlayGraphics uses the static configuration.

Copyright © 2026 Vizrt

5 Using Media Router

This section contains the following:

- Running Media Router as a Console Application
- Running Media Router as a Windows Service
- Media Router Console Commands
- Redundancy Setup
- Mirror Mode

Copyright © 2026 Vizrt

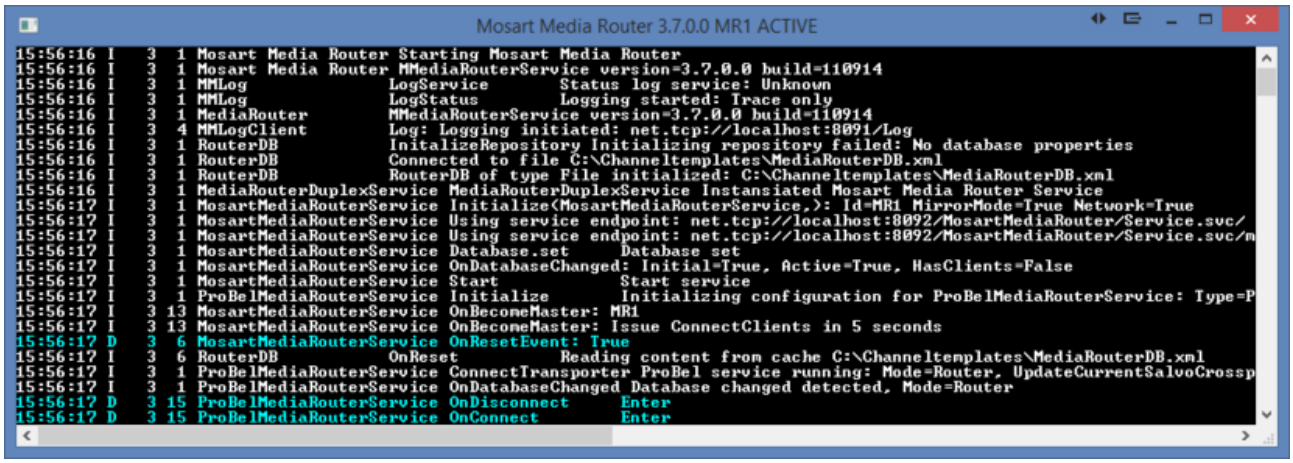
5.1 Running Media Router as a Console Application

Normally Media Router is started as a console application. This makes it possible to monitor changes done by the Media Router as well as controlling it by using appropriate Media Router Console Commands.

```
%ProgramFiles%\Mosart Medialab\Mosart Media Router
Service\MMediaRouterService.exe
```

Note: It is recommended to stop any running Media Router service prior to starting the Media Router as a console application.

Start the application as any other Windows application. The console window will appear as shown in the figure below:



Note that the initial messages in the console window gives information regarding the various protocols (services) used by the Media Router. This according to the Media Router service configuration

In the sample above the following services are configured:

- Media Router Service – Used internally by Viz Mosart applications. Used to communicate with Viz Mosart Server using a proprietary protocol
- ProBel Media Router Serial Service – Allows router control via the ProBel protocol.

Note: When starting the application the first time, the following error message might appear:

```
Failed to initialize media router service of type RestMediaRouterService:
HTTP could not register URL http://+:8094/MosartMediaRouter/Rest.svc/.
Your process does not have access rights to this namespace (see http://
go.microsoft.com/fwlink/?LinkId=70353 for details).
```

This is related to the Service configurations described in section Media Router Service Configuration (Rest Media Router Service) and can be solved in two ways:

Copyright © 2026 Vizrt

- Either execute the `netsh http add urlacl url=http://+:8094/ user=Everyone` command in an elevated command prompt, and restart the Media Router application
  or
- Run the Media Router application as an Administrator.

Copyright © 2026 Vizrt

5.2 Running Media Router as a Windows Service

The Media Router installer automatically registers Media Router as a Windows service.

This means that starting and stopping the Media Router as a service may be done via the Windows Service Manager. The service name is **Mosart Media Router Service**.

> **Note:** It is recommended to stop the Media Router service prior to starting the Media Router as a console application.

Copyright © 2026 Vizrt

5.3 Media Router Console Commands

The Media Router Console enables actions like:

5.3.1 To activate predefined salvos

5.3.2 To set individual crosspoints

5.3.3 To get router information

The following commands are available:

- **autofailover:** Toggles Auto Failover mode. See the MosartMediaRouterService configuration.
- **autotake:** Toggle autotake mode. See the MosartMediaRouterService configuration.
- **clear:** Clears the Console window.
- **command:** Syntax: `command cmd [arg1] [arg2]` . Executes a Media Router command with two optional arguments. The following commands are available:
  - **backup:** Enforces a backup of the Media Router database.
  - **get <filename>** : Retrieves the current Media Router database and stores the database using the given filename.
  - **reset:** Resets the Media Router to the latest Media Router database stored on the file system.
  - **restore <filename>** : Restores the Media Router database to the content of the database stored in the given filename.
  - **flush:** Flushes the current content of the Media Router to file storage. This command may be used to set the initial state of the Media Router.
- **deletesalvo:** Syntax: `deletesalvo salvo` . Deletes the specified salvo.
- **exit:** Exits the Media Router.
- **getconfig:** Syntax: `getconfig client` . Returns the current configuration for the given client. Used mainly for testing Mosart Server clients.
- **getstatus:** Returns an xml string containing current Media Router status.
- **help:** Lists all commands.
- **idle:** Toggles active / idle mode of the Media Router service. Used for testing.
- **list:** Syntax: `list [salvos|clients|sources|inputs|outputs|all]` . List various router information.
- **restore:** Syntax: `restore <filename>` . Restores the Media Router database to the content of the database stored in the given filename.
- **save:** Syntax: `save <filename>` . Retrieves the current Media Router database and stores the database using the given filename.
- **setactive:** Activates the Media Router service.
- **setcrosspoint:** Syntax: `setcrosspoint inport outport [salvo]` . Sets a crosspoint in the given salvo:

Copyright © 2026 Vizrt

    - If no salvo is given then the *pending* salvo is used. A *set crosspoint* request is sent to the corresponding clients.
    - If the given salvo is not present, a *new* salvo will be created.
- **setsalvo:** Syntax: `setsalvo salvo`. Fires the specified salvo.
- **verbose:** Turns on/off verbose diagnostics.

Info: When specifying crosspoints in console commands, use the inport/outport name to identify the corresponding port.

5.3.4 Media Router console command examples

- List all current salvos: `list salvos`
- List active connections: `list salvos current`
- List pending connections: `list salvos pending`
- List a specified salvo: `list salvos M1.A`
- Connects video port M1.A.A to Quantel 2 / Port 1: `setcrosspoint VS.Q2.1 M1.A.VA`
- Sets a crosspoint in a salvo: `setcrosspoint VS.Q2.1 M1.A.VA M1.Backup`
- Fires a salvo: `setsalvo M1.Backup`
- Deletes a salvo: `deletesalvo M1.Backup`

Copyright © 2026 Vizrt

5.4 Redundancy Setup

It is possible to use two Media Router service instances in a redundancy setup. In this case one of the Media Router service instances is configured to be the Primary in a Primary/ Secondary relationship. The following rules are applied:

- There shall be only one Media Router service configured as Primary. This is done using the **PreferredMaster** property.
- The Secondary Media Router service will automatically take control if the Primary is down or not responding. This behaviour may be overridden by setting the **AutoFailover** property to false. If so, activating the secondary Media Router service has to be done manually by issuing the **SetActive** command.
- The Primary Media Router service will always be the active Media Router service instance if it is running. This means that if the Primary is started then the Primary will take over the control of any active secondaries.
- Inactive or idle Media Router services will ignore any crosspoint changes sent from any clients.
- The Primary Media Router service will ensure that the Media Router state is synchronized with the secondary. This way, both Primary and Secondary shall always have the same configuration and crosspoints.
- Clients controlling the Media Router services have to apply one of the following strategies:
    - Send crosspoints to both Primary and Secondary Media Router services.
    - Send crosspoints only to the active Media Router service. In this case the clients need to monitor whether the Media Router services are active or not.

5.4.1 Related service properties

- **Id:** To set an unique identity of the Media Router service
- **PreferredMaster:** Set to true for the primary Media Router service
- **Slave:** Connection string to the other Media Router service in a redundancy setup.
- **AutoFailover:** Set to *false* to ignore automatic failover by the secondary when loosing connection to the primary.

Copyright © 2026 Vizrt

5.5 Mirror Mode

Viz Mosart servers are normally configured in pairs in a redundancy setup. In most cases, configurations sent to both the Viz Mosart main and backup servers should be identical. In this way, both Viz Mosart servers are controlling the same devices.

In Media Router, main/backup redundancy setup requires *both* Viz Mosart servers to be configured as clients, with the same set of outports; and the configurations to both Viz Mosart servers be synchronized.
This requires Media Router to send the same configuration to both.

This is done by activating Mirror Mode

In Mirror Mode setting, a crosspoint related to one Viz Mosart server in a redundancy setup then the same configuration will be sent to both Viz Mosart servers.

5.5.1 Related service properties

- **MirrorMode:** Set to *True* to activate Mirror Mode
- **MirrorModeMaster:** Specifies which Viz Mosart server will receive crosspoint changes.
- **SynchronizedMirroring:** If set, crosspoint changes related to any Viz Mosart server will be reflected to both.

Copyright © 2026 Vizrt

6 MMT Test Application

The Media Router Test Application (MMT) is a console application for testing the Media Router service. It may be installed as part of the Mosart Test Suite (separate installer) and used for testing, verifying and controlling the Media Router.

This section contains the following:

- MMT Console Commands
- MMT Emulating a Viz Mosart Server Client

6.1 To install MMT

- Execute the file *%ProgramFiles%\Mosart Medialab\Mosart Test Suite\MediaRouterTester.exe*

Copyright © 2026 Vizrt

6.2 MMT Console Commands

From Media Router Test Application (MMT) it is possible to activate predefined salvos, to set individual crosspoints and to get router information by entering appropriate console commands. The following commands are available:

- **autotake:** Toggle autotake mode. See the MosartMediaRouterService configuration.
- **clear:** Clears the console window
- **configconnect:** Syntax: configconnect [config]. Connects to an Media Router service using a named predefined configuration located in the MMT application configuration file, MediaRouterTester.exe.config
- **connect:** Syntax: `connect [name] [host] [port]`. Connects to a Media Router service located at host:port using the give name as subscription name. Issuing connect without any parameters is the same as a reconnect.
- **crosspointchanged:** Syntax: `crosspointchanged inport outport [salvo]`. Emulates a crosspoint change from the MMT
- **deletesalvo:** Syntax: `deletesalvo salvo`. Deletes the specified salvo
- **exit:** Exits the Media Router
- **getstatus:** Returns an xml string containing current Media Router status.
- **help:** Lists all commands
- **list:** Syntax: `list [salvos|clients|sources|inputs|outputs|all]`. List various router information
- **listen:** Syntax: `listen name port`. Starts listening for Media Router connection using given name and port. Same as –N and –L command line arguments.
- **restore:** Syntax: `restore <filename>`. Restores the Media Router database to the content of the database stored in the given filename.
- **save:** Syntax: `save <filename>`. Retrieves the current Media Router database and stores the database using the given filename
- **sendcommand:** Syntax: `sendcommand command [arg1] [arg2]`. Sends a command to the Media Router with two optional arguments. See the corresponding Media Router console command for a list of valid commands.
- **sendheartbeat:** Forces a heartbeat to be sent to the connected Media Router service. For testing
- **setactive:** Activates the connected Media Router service
- **setcrosspoint:** Syntax: `setcrosspoint inport outport [salvo]`. Sets a crosspoint in the given salvo:
  - If no salvo is given then the pending salvo is used. I.e. a set crosspoint request is sent to corresponding clients.
  - If the given salvo is not present, a new salvo will be created.
- **setsalvo:** Syntax: `setsalvo salvo`. Fires the specified salvo
- **subscribe:** Syntax: `subscribe <name>`. Changes the subscription name. Used when using the MMT for emulating a Mosart server. See also the connect command
- **swap:** Syntax: `swap <client1> <client2>`. Swaps configuration between client1 and client2. Used in redundancy setup not using mirror mode. Not recommended.
- **verbose:** Turns on/off verbose diagnostics

Copyright © 2026 Vizrt

6.3 MMT Emulating a Viz Mosart Server Client

6.3.1 MMT as a second Viz Mosart server client emulator

To start MMT second Viz Mosart server client in emulator mode

The Media Router is configured to connect to two Viz Mosart Servers:

1. Mosart1 @ <hostname>:8191
2. Mosart2 @ localhost:8192

AV Automation at Viz Mosart Server @ <hostname> is configured as Mosart1:8191.

The second Viz Mosart Server client is used for monitoring the Media Router.

- Start MMT with command arguments "-N Mosart2 –L 8192
  to emulate the second Viz Mosart Server client:
  `%ProgramFiles%\Mosart Medialab\Mosart Test Suite\MediaRouterTester.exe –N Mosart2 –L 8192`

Copyright © 2026 Vizrt

7 Media Router REST Protocol

The REST protocol can be used to communicate with the Viz Mosart Media Router.

The REST protocol is based upon using HTTP GET for all protocols, including setting crosspoints and properties on the server. This enables straightforward testing of all functionalities using the REST protocol with a web browser.

This section covers the following topics:

- Retrieval Commands
  - clients
  - sources
  - inports
  - outports
  - salvos
  - current
  - pending
  - config
  - crosspoints
- Control Commands
  - setcrosspoint
  - setsalvo

7.1 Viz Mosart Media Router REST Command Format

The following command is the general Mosart Media Router REST protocol command:

```
http://[hostname]:[port]/MosartMediaRouter/Rest.svc/<command>?<params>
```

hostname: Server name or IP-address for the Mosart Media Router REST service.

port: TCP/IP port for the Mosart Media Router REST service. Default 8094.

<command>: Command issued.

<params>: Optional parameters for the command.

For simplicity, the service part of the REST URI templates will be replaced with [service] in the following:

`[service] =` http://[hostname]:[port]/MosartMediaRouter/Rest.svc

7.2 REST Commands

The available commands via the REST protocol are as follows:

Copyright © 2026 Vizrt

Command	Description
clients	Retrieves information of all clients.
sources	Retrieves information of all sources.
inports	Retrieves information of all inports or inports connected to a given source
outports	Retrieves information of all outports or outports connected to a given client
salvos	Retrieves information of a single or multiple salvos
current	Retrieves information of the current state of the router
pending	Retrieves information of the pending state of the router
config	Retrieves information of the current or salvo specific configuration for a given client
crosspoints	Retrieves information of changed crosspoints from a given timestamp
setcrosspoint	Sets a specified crosspoint
setsalvo	Fires a specified salvo
status	Returns current Media Router status

To communicate with the Mosart Media Router REST service, the event client requires an authentication process. Depending on who the user is, some parts of the router may be restricted. The specifications for authentication are provided in the Retrieval Commands. The authentication process is based on OAuth, a common protocol for web services for this purpose. Details of this authentication process are still under development.

7.2.1 Retrieval Commands

This section contains commands for obtaining information.

clients

The *clients* command is used to obtain information about the clients of the Mosart Media Router. A typical client is a Mosart Server or Omnibus.

The REST URI template for the clients command:

```
[Service]/clients
```

Copyright © 2026 Vizrt

sources

The *sources* command is used to obtain information about the sources of the Mosart Media Router. A typical source is a video playout server.

The REST URI template for the sources command:

```
[Service]/sources
```

inports

The *inports* command is used to obtain information about the inports of the Mosart Media Router. A typical inport is a port of a video playout server. Two variants of this command exist. One for retrieving all inports and one for retrieving inports for a given source. The REST URI template for the inports command:

```
[Service]/inports
[Service]/inports?source=[source]
```

source: Name of the source to obtain inports for.

outports

The *outports* command is used to obtain information about the outports of the Mosart Media Router. A typical outport is a port used by a Mosart Server. Two variants of this command exist. One for retrieving all outports and one for retrieving outports for a given client. The REST URI template for the outports command:

```
[Service]/outports
[Service]/outports?client=[client]
```

client: Name of the client to obtain outports for.

salvos

The *salvos* command is used to obtain information about the registered salvos within the Mosart Media Router. Two variants of this command exist. One for retrieving information for all salvos and one for retrieving information for a given salvo.

The REST URI template for the salvos command:

```
[Service]/salvos
[Service]/salvos?salvo=[salvo]
```

salvo: Name of the salvo to obtain information for.

Copyright © 2026 Vizrt

current

The *current* command is used to obtain information about the current salvo within the Mosart Media Router. The current salvo is the same as the current state of the Mosart Media Router.

The REST URI template for the current command:

```
[Service]/current
```

pending

The *pending* command is used to obtain information about the pending salvo within the Mosart Media Router. The pending salvo is the same as the current state but may contain crosspoints that are not verified (pending) by the clients.

The REST URI template for the pending command:

```
[Service]/pending
```

config

The *config* command is used to obtain information about the crosspoints for a given client either for the current or a specified

The REST URI template for the config command:

```
[Service]/config?client=[client]
[Service]/config?client=[client]&salvo=[salvo]
```

client: Name of the client to obtain the configuration for.

salvo: Name of the salvo to obtain the client configuration for, if specified.

crosspoints

The *crosspoints* command is used to obtain information about the crosspoint that have been changed since a given timestamp. This command is normally used to poll the Mosart Media Router for its current state.

The REST URI template for the crosspoints command:

```
[Service]/crosspoints?from=[timestamp]
```

timestamp: Returns all crosspoints changed after the specified timestamp.

Copyright © 2026 Vizrt

7.2.2 Control Commands

Commands in this section control the Mosart Media Router, by changing the state of the Mosart Media Router.

> **Info:** The HTTP GET protocol is used instead of HTTP PUT. This makes it possible to test the commands using an ordinary web browser.

setcrosspoint

The *setcrosspoint* command is used to set a crosspoint in any salvo. If no salvo is given, the crosspoint is set in the pending salvo to be executed immediately. If the specified salvo does not exist, a new salvo with the corresponding name is generated.

The REST URI template for the *setcrosspoint* command:

```
[Service]/setcrosspoint/[inport]/[outport]
[Service]/setcrosspoint/[inport]/[outport]?salvo=[salvo]
```

inport: Specifies the inport of the crosspoint.

outport: Specifies the outport of the crosspoint.

salvo: Specifies the salvo where the crosspoint shall be set. If no salvo is given, the crosspoint is set directly in the router (pending salvo).

setsalvo

The *setsalvo* command is used to fire a salvo.

The REST URI template for the setsalvo command:

```
[Service]setsalvo/[salvo]
```

salvo: Specifies the salvo to fire.

Copyright © 2026 Vizrt

8 MMR Configuration of Vizrt Graphics

Note: MMR is **not** currently supported with the new graphics driver that was introduced with *Viz Mosart 5.3*.

- Configuration of Viz MSE
- Configuration of Viz Engine
- Configuration of Mosart Fullscreen Graphics
    - Mosart Server (as an MMR client) properties
- Configuration of Mosart overlay graphics
    - Mosart Server Overlay (outport) Properties
- Viz Graphics Additional Properties

All MMR configuration is performed directly in an xml file, `MediaRouterDB.xml`.

Note: `MediaRouterDB.xml`, is updated when using MMR with state information. It is therefore recommended to keep the original configuration in a separate file and copy any changes to the file used by MMR.

You specify the Viz Media Sequencer (MSE) and Viz Engines to be used by a Mosart installation in the MMR configuration as *sources* and *inports* respectively.
An *inport* shall always be attached to a single client which means for Viz graphics all Viz Engines must be associated with a single Viz MSE. This means that a configuration where a Viz Engine is used by *different* Vizrt MSEs, the Viz Engine must be specified multiple times, one for each MSE.

Below are some examples how Viz Graphics is configured in the MMR configuration file, `MediaRouterDb.xml`. Note that only details relevant to this example are shown.

8.1 Configuration of Viz MSE

Viz MSE is configured as a graphics controller and in MMR terms, configured as a *source*:

```
<Sources>
    <Value>
        <item id = "FG.NEWS" ignore = "false" type = "Vizrt" category =
"FullscreenGraphics">
            <ConnectionString>Server=hostname</ConnectionString>
        </item>
    </Value>
</Sources>
```

where:

- **Id:** Unique identifier for the *source*. Used in *inports* to relate to the *source*. Recommended to follow naming convention in the MMR database configuration.
- **ignore:** If set to true, the *source* will be ignored by MMR. Used to disable configuration.

Copyright © 2026 Vizrt

- **type:** Graphics type. Set to “Vizrt” for Viz Graphics
- **category:** Device category. Set to either *FullscreenGraphics* or *OverlayGraphics*. Both types can be used for both FullscreenGraphics and OverlayGraphics.
- **ConnectionString:** *Source* specific properties, separated by ‘;’. For Viz MSE:
    - Syntax: Server=hostname[:port][,hostname[:port]][;AutoHandlerNames=true]
    - **Server:** Specify hostname/ip-address of Viz MSE:
        - **Syntax:** Server=hostname[:port][,hostname[:port]]
        - Some examples:
            - Backup MSE: main_host,backup_host
            - Tcp/ip port: hostname:port
    - **AutoHandlerNames:** To generate Vizrt MSE handler names automatically.
        - Syntax: AutoHandlerNames=[true,false]
        - Default true for Viz Graphics
        - AutoHandlerNames = true is required for mirrored graphics.

8.2 Configuration of Viz Engine

Viz Engines are configured in MMR terms as an *inport* and therefore must be related to a *source:*

```
<Inports>
   <Value>
      <item id="FG.NEWS.E1" order="1" ignore="false" device="FG.NEWS" deviceName="1"
            category="FullscreenGraphics" deviceType="Vizrt" ">
         <ConnectionString>Host=hostname; Encoding=UTF-8</ConnectionString>
      </item>>
   </Value>
</Inports>
```

where:

- **Id :** Unique identifier for the *inport*. Recommended to follow naming convention in the MMR database configuration.
- ***order:*** Used to relate to Probel crosspoint source
- **ignore:** If set to true, the *inport* will be ignored by MMR. Used to disable configuration.
- **device:** Specifies the *source* the inport belongs to. A *source* in this context refers to a Viz MSE. Must be a valid *source*.
- **deviceName:** Device specific. For Viz graphics this is used to specify the viz handler name as used by the Viz MSE. This is ignored when *AutoHandlerNames* = true (see configuration of Viz MSE)
- **type:** Graphics type. Set to “*Vizrt*” for Viz Graphics
- **category:** Device category. Set to either *FullscreenGraphics* or *OverlayGraphics*. Both types can be used for both FullscreenGraphics and OverlayGraphics.
- **ConnectionString:** Device specific properties, separated by ‘;’. For Viz Engine:
    - Syntax: hostname[:port][,hostname[:port]][,hostname[:port][,…][;Method=scene_AB_mode][;Encoding=UTF-8]
    - **Host=hostname:** Specify hostname(s) for the Viz Engine
        - Syntax: hostname[:port][,hostname[:port]][,hostname[:port][,…]

Copyright © 2026 Vizrt

- Use multiple hostnames for mirrored graphics.
  When using a *single* hostname, mirroring can be achieved by configuring the *deviceName* property on the outports as described in Mosart Server Fullscreen (outport) Properties and Mosart Server Overlay (outport) Properties below.
- Optional TCP/IP ports are specified by appending the port after '**:**'
- Some examples:
  - Mirrored graphics: hostname1,hostname2
  - TCP/IP port: hostname:6110

- Method=scene_AB_mode – Used to specify scene transition mode on Viz Engine.
  - Syntax: Method=scene_AB_mode
- Encoding – sets the MSE encoding. Normally UTF-8

8.3 Configuration of Mosart Fullscreen Graphics

Mosart fullscreen graphics are configured in MMR terms as an *outport* related to a *client* which specifies a Mosart server.

```
<Clients>
   <Value>
      <item id="STUDIO1" ignore="false" type="Mosart">
         <ConnectionString>Server=hostname;Port=8099</ConnectionString>
      </item>
   </Value>
<Clients>

<Outports>
   <Value>
      <item id="STUDIO1.FG1" order="1" ignore="false" device="STUDIO1" deviceName="1"
 category="FullscreenGraphics" deviceType="Mosart" />
   </Value>
</Outports>
```

The *client* is used to specify a Mosart Server. The connection point for MMR is AV Automation. To enable MMR on a Mosart Server is therefore done in AV Automation.

8.3.1 Mosart Server (as an MMR client) properties

The Mosart Server itself is configured as an MMR *client* with the following properties:

- *Id* – Unique identifier for the *client*. Used in *outports* to relate to the *client*. Recommended to follow naming convention in the MMR database configuration.
- *ignore* – If set to true, the *client* will be ignored by MMR. Used to disable configuration.
- *type* – Identifies client type. Set to "Mosart"
- *ConnectionString* – Sets the connectivity properties for the Mosart Server
  - Server=hostname – Specifies the server hostname or ip-address for the Mosart Server

Copyright © 2026 Vizrt

- Port=portno – Specifies the tcp/ip listening port used by AV Automation in the Mosart Server. This port is configurable in AV Automation in Devices / Preferences / General tab / MediaRouter section.

8.3.2 Mosart Server Fullscreen (outport) Properties

Mosart Server fullscreen graphics is configured as MMR *outports* with the following properties:

- ***Id:*** Unique identifier for the *outport*. Recommended to follow naming convention in the MMR database configuration.
- **order:** Relates to the Probel crosspoint destination.
- **ignore:** If set to true, the *outport* will be ignored by MMR. Used to disable configuration.
- **device:** Specifies the *client* the *outport* belongs to. Must be a valid *client*.
- **deviceName:** For Viz fullscreen graphics, this value specifies the corresponding fullscreen engine used by AV Automation.
  Must be a number in the range [*1..5*] or [*11..15*] where the latter is mirrored engines in the following pairs: (*1,11*), (*2,12*), …, (*5,15*).

> **Note:** Mirroring for Viz graphics is also possible by specifying multiple hostnames in the ConnectionString for the Viz Engine.

- **category:** Device category. Set to *FullscreenGraphics* for fullscreen graphics.

8.4 Configuration of Mosart overlay graphics

Mosart overlay graphics are configured in MMR terms as an *outport* related to a *client* which specifies a Mosart server.

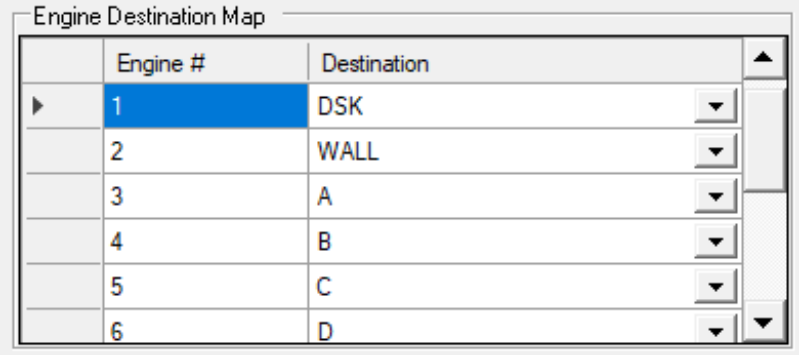
```
<Clients>
   <Value>
      <item id="STUDIO1" ignore="false" type="Mosart">
         <ConnectionString>Server=hostname;Port=8099</ConnectionString>
      </item>
   </Value>
<Clients>

<Outports>
   <Value>
      <item id="STUDIO1.OG1" order="2" ignore="false" device="STUDIO1"
deviceName="DSK" category="OverlayGraphics" deviceType="Mosart" />
   </Value>
</Outports>
```

8.4.1 Mosart Server Overlay (outport) Properties

Mosart Server overlay graphics are configured as MMR *outports* with the following properties:

Copyright © 2026 Vizrt

- **Id:** Unique identifier for the *outport*. Recommended to follow naming convention in the MMR database configuration.
- **order:** Used to relate to Probel crosspoint destination.
- **ignore:** Used to disable configuration. If set to true, the *outport* will be ignored by MMR.
- **device:** Specifies the *client* the *outport* belongs to. Must be a valid *client*.
- **deviceName:** For Viz overlay graphics, specifies a graphics destination.
    - Option #1: Enter a graphics destination name. Using the *same* device name for multiple *outports* turns on mirroring for the destination.
    - Option #2: Enter an Engine number. This is the legacy method, requiring mapping between Engine number and destination.
      The destination names must be present in the static configuration.
      This mapping table is part of the OverlayGraphics static configuration (**Mosart Graphics Interface > Settings > Properties > MMR** tab).



Note: Mirroring for Viz graphics is also possible by specifying multiple hostnames in the ConnectionString for the Viz Engine.

- **category:** Device category. Set to *OverlayGraphics* for overlay graphics.

8.5 Viz Graphics Additional Properties

From the configuration there are some properties that are important when using MMR and mirrored graphics:

8.5.1 Apply Channel Name to Elements

This property is available in both OverlayGraphics and FullscreenGraphics static configurations.
When enabled all graphics elements will be tagged with the corresponding destinations in the Mosart playlists. The graphics will also be controlled via the Viz MSE Channels. This is the preferred and recommended way by Viz MSE. Enabling this property is a requirement for *mirrored* graphics.

For MMR, this property is set using the **ApplyChannelNameToNewElements** property located in the *GraphicsDefaultConfiguration.xml* configuration file.

Copyright © 2026 Vizrt

8.5.2 Remove Unused Channels Outputs

This property is available in the *VizrtGraphicsConfiguration.xml* configuration file. When enabled, the Mosart profile in the MSE will synchronize to the current Mosart fullscreen and overlay configurations.
This is *mandatory* when using MMR.

Copyright © 2026 Vizrt