Viz Arc User Guide

Version 2.2



Copyright ©2025 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. his 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

2025/09/12

Contents

1 Introduction 9
1.1 Related Documents 9
1.2 Feedback and Suggestions 10
2 Getting Started 11
2.1 System Overview 12
2.1.1 Ports 12
2.2 System Requirements 15
2.2.1 Minimum Software Requirements 15
2.2.2 Minimum Hardware Requirements 15
2.2.3 Antivirus 15
3 Installation 17
3.1 Installing Viz Arc 18
3.2 WIBU-based Licensing System 20
3.2.1 Important Pre-installation Information 20
3.3 Launching Viz Arc 21
3.3.1 Command Line Arguments 22
3.4 License 23
3.4.1 Viz Arc Core 23
3.4.2 Viz Arc Freemium 23
3.4.3 Viz Arc REST 23
3.4.4 License Locations 24
3.4.5 License Configuration 24
3.5 Files and Folders 27
3.6 Unreal Plug-in 29
3.7 Engine Service 31
3.7.1 Prerequisites 31
3.7.2 Installation 31
3.7.3 Manual Installation 33
3.7.4 Starting Engine Service 33
3.7.5 Configuration 34
3.7.6 Troubleshooting 35
3.8 Configuring Graphic Hub 37
3.8.1 Configuring Graphic Hub for the Viz Arc Template Workflow 37

3.8.2 Import an Archive Containing Add-ons 37
3.8.3 Manually Creating Add-ons 38
3.9 Script Debugging 45
3.9.1 How to Enable Viz Arc Script Debugger on Windows 7/8/8.1/10 45
4 Configuration 47
4.1 General Configuration 48
4.2 General Settings 49
4.2.1 System 49
4.2.2 Communication 49
4.2.3 General 50
4.2.4 Playlist 51
4.2.5 Scripting 51
4.2.6 Project Folder 51
4.2.7 Unreal Engine 51
4.2.8 Map 52
4.2.9 Toast Notifications 52
4.2.10 Performance 52
4.2.11 User Interface Appearance 53
4.2.12 Default Color Setting 55
4.2.13 Configuring Viz Engine 57
4.3 Profiles 59
4.3.1 Profile Configuration 61
4.3.2 Engine Configuration 62
4.3.3 Channel Configuration 64
4.4 Keyboard Shortcuts 70
4.4.1 Keyboard Shortcuts 70
4.4.2 To Manage Shortcuts 73
4.4.3 To Add Custom Action Shortcuts 73
4.5 Video Settings 76
4.5.1 Preview Output 76
4.6 GPI Setup 83
4.6.1 GPI 83
4.6.2 Triggers 83
4.7 Timecode Setup 85
4.7.1 Time Code 85
4.8 Clip Setup 86
4.8.1 Coder Preset 86

4.8.2 Viz Recorder Preset 88
4.9 Info 92
4.10 Tracking 94
4.10.1 Tracking 95
4.10.2 General 95
4.10.3 SMURF Input 95
4.10.4 Tracking Hub 95
4.10.5 Performance 96
4.10.6 Appearance 96
4.10.7 Graphics 97
4.11 Vizrt Systems 98
4.11.1 Graphic Hub REST 98
4.11.2 Media Service 99
4.11.3 Viz One 100
4.11.4 Viz Virtual Studio 100
4.11.5 TCP Command Port 102
4.11.6 Director 102
4.12 Third Party Systems 108
4.12.1 Vinten Automation Server 109
4.12.2 Tecnopoint Automation Server 109
4.12.3 Telemetrics Automation Server 109
4.12.4 Utah Scientific Router 110
4.12.5 Monogram 110
4.12.6 MIDI Device 112
4.12.7 Stream Deck 113
4.12.8 Art-Net DMX 113
4.13 Loupedeck Integration 114
4.13.1 Installation 114
4.14 Stream Deck Integration 119
4.14.1 Installation 120
4.14.2 Configuration 121
4.14.3 Internal vs. External Stream Deck Control 124
4.14.4 Usage 124
4.15 Preview Studio Configuration 125
4.15.1 Overview 125
4.15.2 Configuration 126
5 Working with Viz Arc 128

5.1 General Workflow 129
5.2 User Interface 131
5.2.1 Builder Mode 131
5.2.2 On Air Mode 135
5.2.3 Top Tool Bar 135
5.2.4 Status Bar 136
5.2.5 Main Menu and Project Toolbar 138
5.2.6 Operations Toolbar 142
5.2.7 Scenes Toolbar 143
5.2.8 Import 144
5.2.9 Export 150
5.3 4 Point Calibration 153
5.3.1 About 4 Point Calibration 153
5.3.2 Prerequisites 153
5.3.3 To Add Calibration Points 154
5.3.4 Calibration 155
5.3.5 Applying Calibration 156
5.4 Supported Action Types 159
5.4.1 Project Action Types 160
5.4.2 Scene Action Types 188
5.4.3 Other Action Types 221
5.5 Builder Mode 241
5.5.1 Actions View 242
5.5.2 Set View 263
5.5.3 Script View 279
5.5.4 Scenes Panel 346
5.6 APIs 357
5.6.1 Viz Arc In-App Web API 358
5.6.2 REST API 379
5.7 Data Integration 387
5.7.1 Singal-R 387
5.7.2 MQTT Broker 388
5.7.3 TCP Server 390
5.7.4 UDP Server 391
5.7.5 DataMap Linking 391
5.7.6 DataMap Websocket 396
5.7.7 Excel and CSV integration 397

5.8 Integrations 402
5.8.1 Art-Net DMX Integration 403
5.8.2 Loupedeck 416
5.8.3 MOS Integration 429
5.8.4 Stream Deck 434
5.8.5 Viz Arena 447
5.8.6 Working with Unreal Engine 460
5.8.7 Timecode from Plura or NDI 500
5.9 Engine Status Widget 505
5.9.1 To Add an Engine Status Widget 505
5.10 Graphic Hub Browser 507
5.10.1 Main Features 508
5.11 Import and Export 515
5.11.1 Import Actions 515
5.11.2 Export Actions 515
5.12 Multizone Chroma Keying 516
5.12.1 Configuring the Multizone Chroma Keyer 516
5.12.2 Working with Polygons and the Map 526
5.12.3 Multizone Chroma Key Action in Detail 529
5.13 OCR 532
5.13.1 To Add an OCR Widget 532
5.14 On Air Mode 547
5.14.1 To Go On Air 547
5.14.2 On Air Mode Desktop Shortcut and Command Line Switches 548
5.14.3 Builder-only Tabs 548
5.15 Playlist 549
5.15.1 Adding Actions to the Playlist 549
5.15.2 Working with the Playlist 550
5.15.3 Using the Playlist 552
5.16 Preview Studio 555
5.16.1 Recording 555
5.16.2 Playback 556
5.16.3 Limitations 558
5.17 Preview 559
5.18 Precision Keyer 561
5.18.1 Simple 561
5.18.2 Advanced 562

5.18.3 Color Correction 562
5.18.4 Picking 563
5.18.5 Settings 565
5.18.6 Presets 565
5.18.7 Execution 566
5.18.8 Web View 566
5.19 Program and Preview 569
5.19.1 Previewing by Keyboard Shortcut 569
5.19.2 Previewing by CTRL Modifier 569
5.19.3 Templates 570
5.20 Projects 571
5.21 Templates 572
5.21.1 Creating a New Template 572
5.21.2 Using a Template 577
5.21.3 Adding Templates to a Master Scene 582
5.21.4 Using Templates on a Scene Containing Multiple Top Level Control Objects 588
5.21.5 Unreal Templates 591
5.21.6 Flowics 596
6 Troubleshooting 613
6.1 Vizrt Issues 613
6.1.1 Unable to Load Scene 613
6.1.2 Graphic Hub Not Connected 614
6.1.3 Preview Not Connected 615
6.1.4 Output Initialize Failed 615
6.1.5 Cannot Connect/Send to Viz Engine 616
6.1.6 Cannot Send Shared Memory Commands 616
6.2 Unreal Engine 617
6.2.1 Unreal Engine not Available 617
6.2.2 Cannot Connect/Send to Unreal Engine 617
6.2.3 Unable to Load Unreal Level 618
6.3 NDI 619
6.4 Support 619

1 Introduction



Viz Arc integrates to Viz Engine and scenes allowing you to design your virtual studios and augmented reality elements in Viz Artist, eliminating the need for creating custom applications to control virtual studios and augmented reality elements for each production.

- A simple user interface allows you to create projects for each production.
- Easily configure action buttons that control any property, element or animation in a Vizrt Scene without any programming or expert skills. Operators can access a panel of actions that can be triggered at any time.
- Arc is an out-of-the-box solution that eliminates the need to create custom controllers for operating AR/VR elements and non-scripted shows.
- Viz Arc lets you easily configure the logic of each individual element and operate virtual studios and augmented reality or graphic elements on a non-scripted live production, while reducing the setup time and expertise required.

1.1 Related Documents

- *Viz Artist User Guide:* Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.
- *Viz Engine Administrator Guide:* Contains information on how to install and configure Viz Artist and Engine, and supported hardware.
- *Viz Graphic Hub Administrator Guide:* Contains information needed to configure and operate Graphic Hub.

Copyright © 2025 Vizrt

- *Viz Tracking Hub Administrator Guide:* Contains information required to understand Tracking Hub and Studio Manager, two important software components in the Viz Virtual Studio solution.
- *Viz Pilot User Guide:* Contains information needed to operate Viz Pilot and use the Newsroom Component to create newsroom data elements for a newsroom system playlist.

1.2 Feedback and Suggestions

We encourage feedback on our products and documentation. Please contact your local Vizrt customer support team at www.vizrt.com.

Copyright © 2025 Vizrt

2 Getting Started

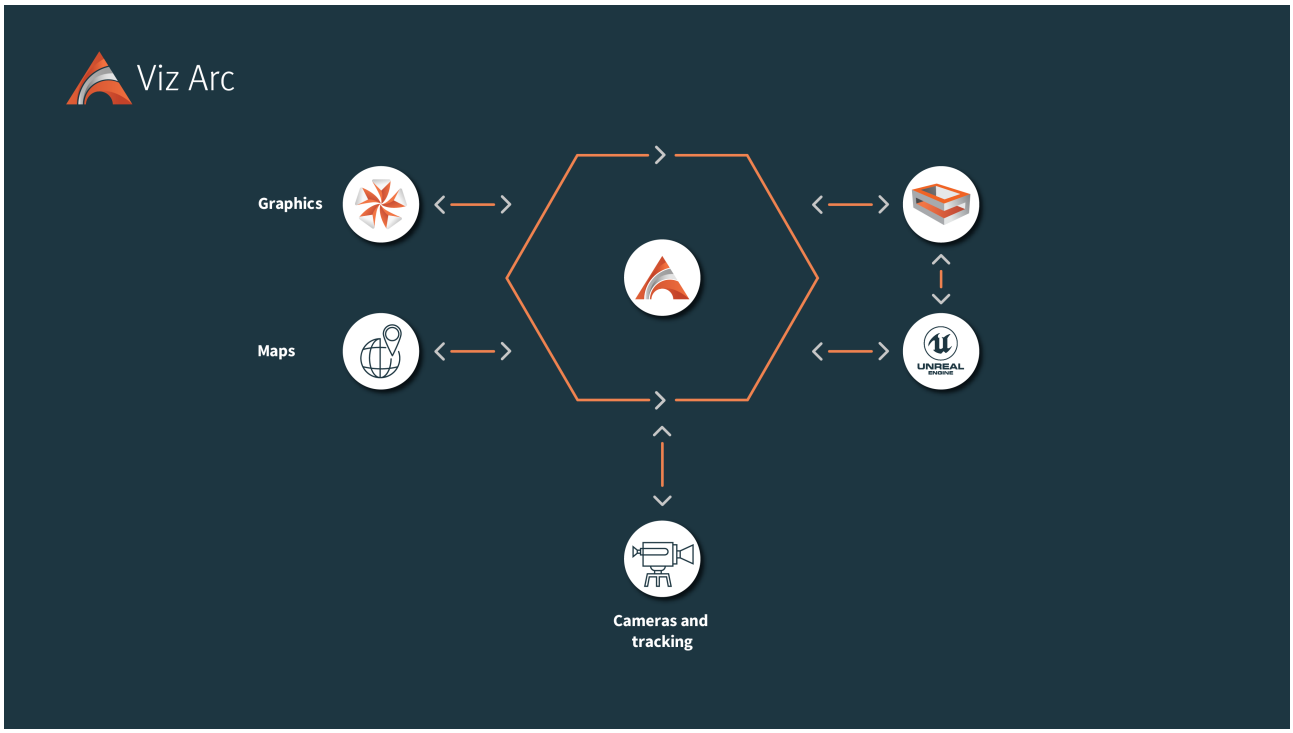
This section describes the location of the installation files, starting and shutting down Viz Arc, and logging on to a Graphic Hub.

This section covers the following topics:

- System Overview
- System Requirements

Copyright © 2025 Vizrt

2.1 System Overview



Viz Arc is a control application that can be used for one or more Viz Engines. Viz Arc uses Graphic Hub and its REST API for browsing Scenes, Images, merged Geometries and Materials.

When working with tracked cameras in a Virtual Studio environment, Viz Arc communicates with Tracking Hub to send and receive camera tracking information. One of the key features of Viz Arc is the ability to set up Augmented Reality elements in outdoor productions directly over a map. A working internet connection is required to retrieve map imagery from providers such as Bing or OSM.

2.1.1 Ports

Application	Port	Protocol
Viz Arc Web Service	Default port: `5004` Default port: `5005`	HTTP HTTPS
Viz Arc REST Service (standalone service)	Default port: `9004` Default port: `9005`	HTTP HTTPS
Viz Arc MQTT Broker	Default port: `1883`	TCP

Copyright © 2025 Vizrt

Application	Port	Protocol
Vic Arc TCP Server	Default port: `9204`	TCP
Viz Arc Template Script Viz Arc Main Script	Debugging port: `9222` Debugging port: `9223`	TCP TCP
Viz Engine	Default ports: `6100` , `6800` Feedback port: `7476`	TCP UDP
Viz Arc Feedback	Feedback port: `12050`	UDP
Graphic Hub REST	REST API port: `19398`	HTTP
Tracking Hub	Requests data through port: `20000` Receives data through port: `4000`	TCP UDP
Media Sequencer	REST API port: `8580`	HTTP
Media Service	for v4.1.0 and higher: `25506` for v2.3.3 and lower: `21099`	HTTP
Pilot Data Server	REST API port: `8177`	HTTP
Coder	Coder port: `8081`	HTTP
Unreal Engine	arcCom port: `5647`	TCP
Engine Service	VizArcEngineService port: `5644`	TCP
Unreal Web Interface	Default port: `30010`	HTTP
Unreal Web Socket	Default port: `30020`	TCP
Vinten Automation Server	Vinten port: `11239`	TCP
Viz One	REST API port: `80`	HTTP

Copyright © 2025 Vizrt

Application	Port	Protocol
Viz Arena	Web API port: `40055`	HTTP
Viz Object Tracker	Default port: `10005`	HTTP
Art-Net Integration	Default port: `6454`	TCP
Monogram Integration	Default port: `51234`	UDP

Copyright © 2025 Vizrt

2.2 System Requirements

This page describes the minimum software and hardware requirements for Viz Arc.

2.2.1 Minimum Software Requirements

The following are minimum software and version requirements for Viz Arc to function properly:

General

- Windows 10/11
- Microsoft .NET Core Framework 8.0
- CodeMeter version 8.00 and later

Vizrt Software

- Viz Engine/Artist 3.14.x: Must be installed as a Video machine and shared memory output must be configured for built-in Viz Output to function.
- Graphic Hub 3.5.0
- Graphic Hub REST 2.5.0

> **Note:** To integrate with Viz Virtual Studio, Tracking Hub 1.1 or higher must be installed.

Epic Games Unreal Engine

- Unreal Engine 5.1 or higher is required for Unreal integration.

2.2.2 Minimum Hardware Requirements

Viz Arc runs on the following platforms:

- Windows 10/11 (64-bit only)
- Windows Server 2012 R2

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

The following prerequisites apply to all platforms. Applying the changes may require local administrator access rights, new or amended group policy entries, or modified services. Contact your local IT manager for assistance.

Please refer to the relevant Viz Engine version hardware requirements.

2.2.3 Antivirus

Vizrt does not recommend or test antivirus systems in combination with Vizrt products, as the use of such systems can potentially lead to performance losses. The decision for the use of antivirus software and thus the risk of impairments of the system is solely at the customer's own risk.

Copyright © 2025 Vizrt

There are general best-practice solutions, these include setting the antivirus software to not scan the systems during operating hours and that the Vizrt components, as well as drives on which clips and data are stored, are excluded from their scans (as previously stated, these measures cannot be guaranteed).

Services and folders which should be excluded from scanning are:

- All the files contained in *C:\Program Files\Vizrt\Viz Arc x.x* (where x.x is the version of Viz Arc).
- All the files contained in *C:\programdata\vizrt\vizarc*.

Copyright © 2025 Vizrt

3 Installation

Before installing the software, make sure your computer meets the System Requirements for hardware and software and that you have installed all related Vizrt software.

This section covers the following topics and procedures:

- Installing Viz Arc
- WIBU-based Licensing System
- Launching Viz Arc
- License
- Files and Folders
- Unreal Plug-in
- Engine Service
- Configuring Graphic Hub
- Script Debugging

Copyright © 2025 Vizrt

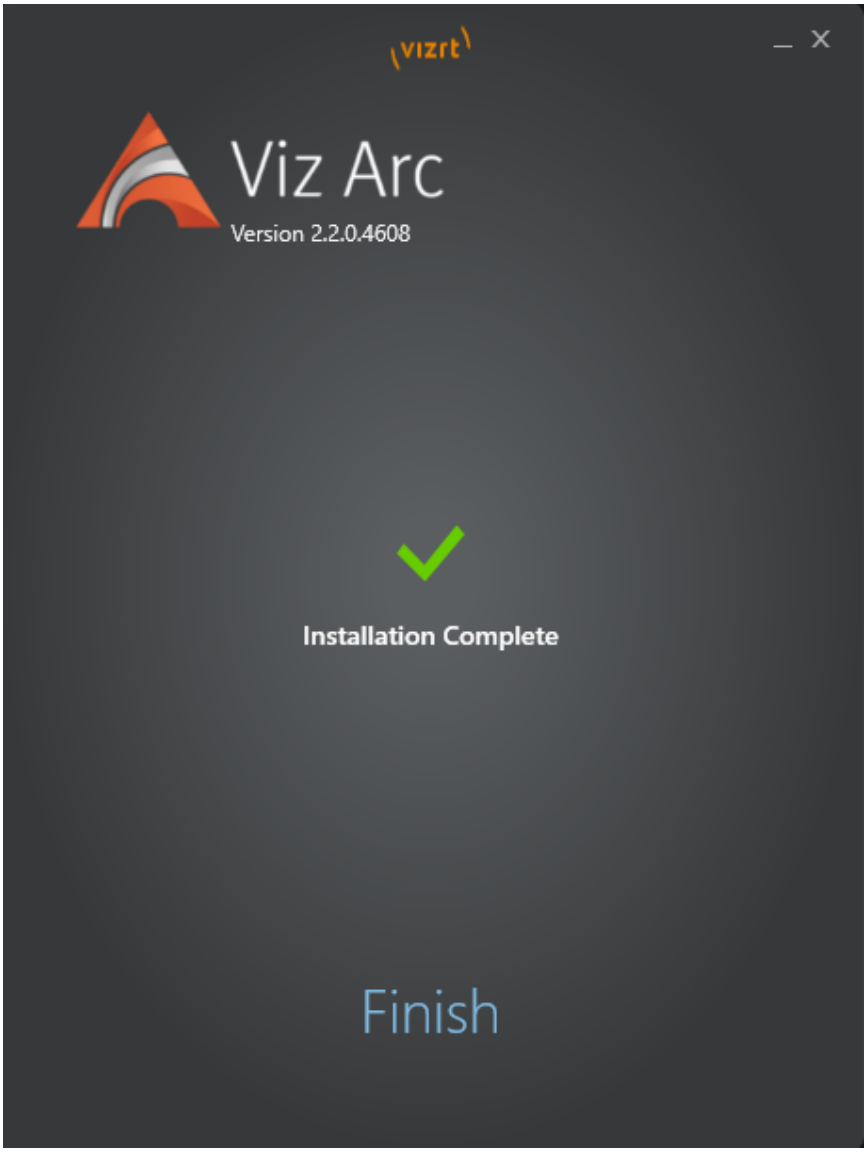
3.1 Installing Viz Arc

1. **Uninstall** any previously installed Viz Arc versions.



2. Select the parts you want to install on the target machine:
    a. **Viz Arc**, the main application.
    b. **Viz Arc REST Service**, required when Viz Arc actions shall be called from Viz Pilot, Pilot Edge or the MOS integration.
    c. **Viz Arc Engine Service**, required on machines running an Unreal Engine which are controlled by Viz Arc. This installer contains the Viz Arc communication plugins for the Unreal Engine which can be found in *C:\ProgramData\vizrt\VizArc\Resources\unreal* after installation. When the installer detects an existing Unreal installation it offers the automatic deployment of the plug-in when the respective checkbox is selected.

Copyright © 2025 Vizrt



Note: After installation, you must first configure your system, and request and apply the license to use all of the software features.

Warning: Installing Viz Arc 2.2.0 over previous installed versions, might lead to unexpected and erratic behavior. Uninstall any previously installed version.

Copyright © 2025 Vizrt

3.2 WIBU-based Licensing System

This chapter describes management and usage of the licensing system based on CodeMeter from WIBU Systems available in Viz Arc 1.0 and later.

Important: Any change in the license configuration requires a restart of Viz Arc.

3.2.1 Important Pre-installation Information

The WIBU licensing system requires the installation of the CodeMeter Runtime Software (included in the Bundle installer):

- 8.00 and newer

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

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

Info: On network disconnect and reconnect, it may happen that a license is checked out twice. Should that occur, release the license manually on the CodeMeter service on the license server or restart the license server.

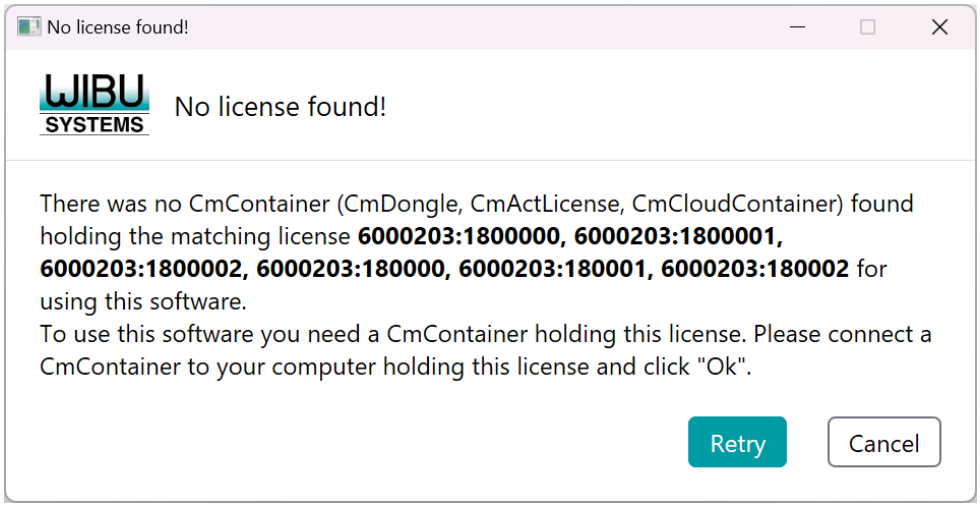
See Also

- Vizrt Licensing Administrator Guide

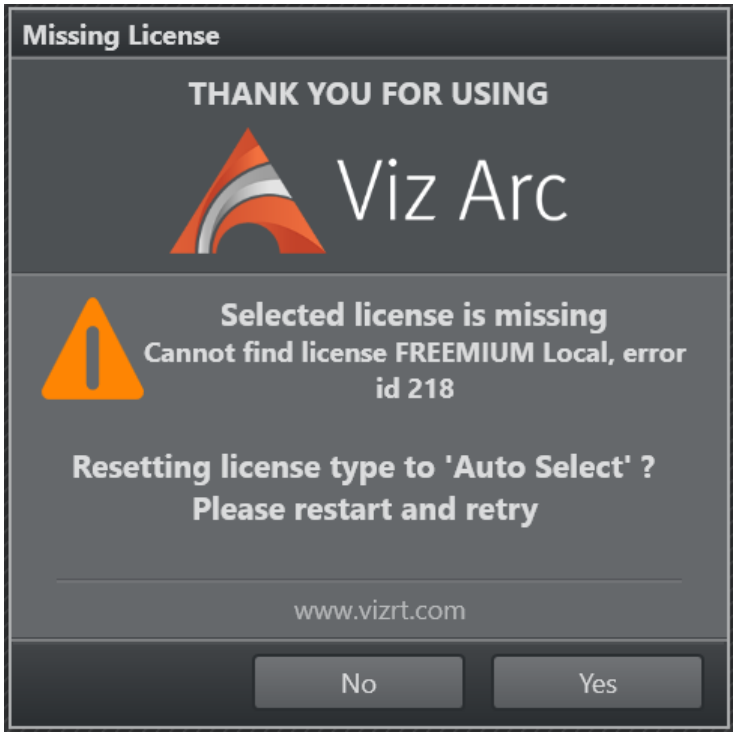
Copyright © 2025 Vizrt

3.3 Launching Viz Arc

Once you've installed Viz Arc, you're ready to start using it! If you don't have a license yet and the system cannot find any compatible license, the following (or similar) message appears:



In case a specific license has been selected but it could not be found, the following dialog appears.



Select **Yes** to reset the license type to **Auto**, then start Viz Arc again. When the application starts you can try again to select a specific license.

Select **No** if you don't want to change the configuration. See the Licensing section for more information.

Copyright © 2025 Vizrt

3.3.1 Command Line Arguments

The following command line switches can be used:

Switch	Description
`-o, --onair`	Start Viz Arc in On Air Mode.
`--project name`	The name of the project to be loaded at startup. Use prefix "local:" or "file:" to explicitly specify a project form the local file system or use the prefix "gh:" or "graphichub:" to specify a project stored on Graphic Hub.
`--projectgh name`	The name of the project on Graphic Hub to be loaded at startup.
`--programdatapath path`	Specify the Viz Arc ProgramData Folder (for example, *C:\ProgramData\vizrt\VizArc*).
`--noproject`	Force startup without any project loaded (default is to open the last opened project).
`-t, --tracking`	Start Viz Arc in Tracking Mode.
`-a, --arena`	Start Viz Arc in Arena Mode.
`--help`	Display help when started from a console.

Examples

`VizArc.exe --project gh:VirtualSetA` : Starts Viz Arc loading the project *VirtualSetA* from Graphic Hub.

`VizArc.exe --project local:ObjectTracker -t` : Starts Viz Arc loading the project *ObjectTracker* form the local file system and it starts up with the Object Tracker user interface already opened.

Copyright © 2025 Vizrt

3.4 License

There are three types of licenses available for Viz Arc:

Features	Quantity
Viz Arc Core	1
Viz Arc Freemium	1
Viz Arc Rest	1

3.4.1 Viz Arc Core

This is the standard license, allowing you to work unlimited with Viz Arc.

3.4.2 Viz Arc Freemium

This license allows you to work with Viz Arc, but does not allow you to save or load actions other than the **Chroma Key Action**. The license is intended to be used by users that need to control the internal **Precision Keyer** of the Viz Engine.

3.4.3 Viz Arc REST

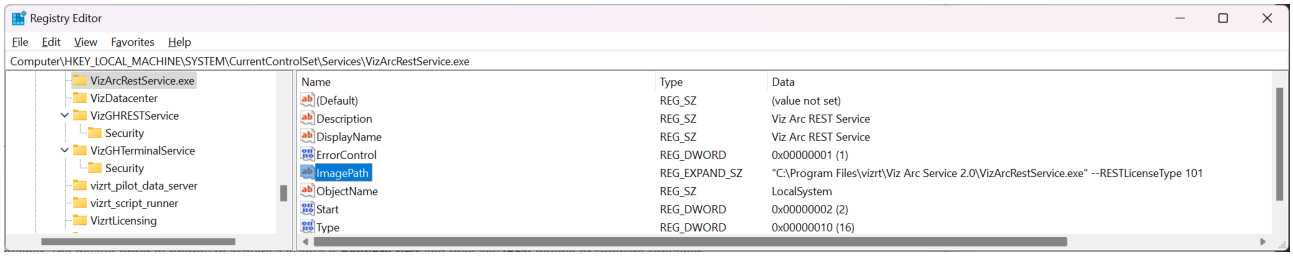
This license is not mandatory for the standard operations of Viz Arc. It is intended for users that need to trigger actions from external applications like **Viz Pilot** or **Viz Mosart**. It is required by the **Viz Arc REST service** that is an additional package included in the bundle installer. The service runs independently from any opened **Viz Arc** instance and allows the user to trigger actions from any project at any time.

By default, the REST server tries to acquire a license based on the Viz Arc's desktop application license selection (if a local license is configured, the REST service looks for a local license as well). The license location for the REST server can be forced using a command line switch `--RESTLicenseType`. Valid values are:

- **0** = Network - Local
- **1** = Local - Network
- **2** = Network
- **3** = Local
- **101** = Network - Local v3
- **102** = Local - Network v3
- **103** = Network v3
- **104** = Local v3

The switch can be added to the registry entry
under *Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VizArcRestService.exe:*

Copyright © 2025 Vizrt

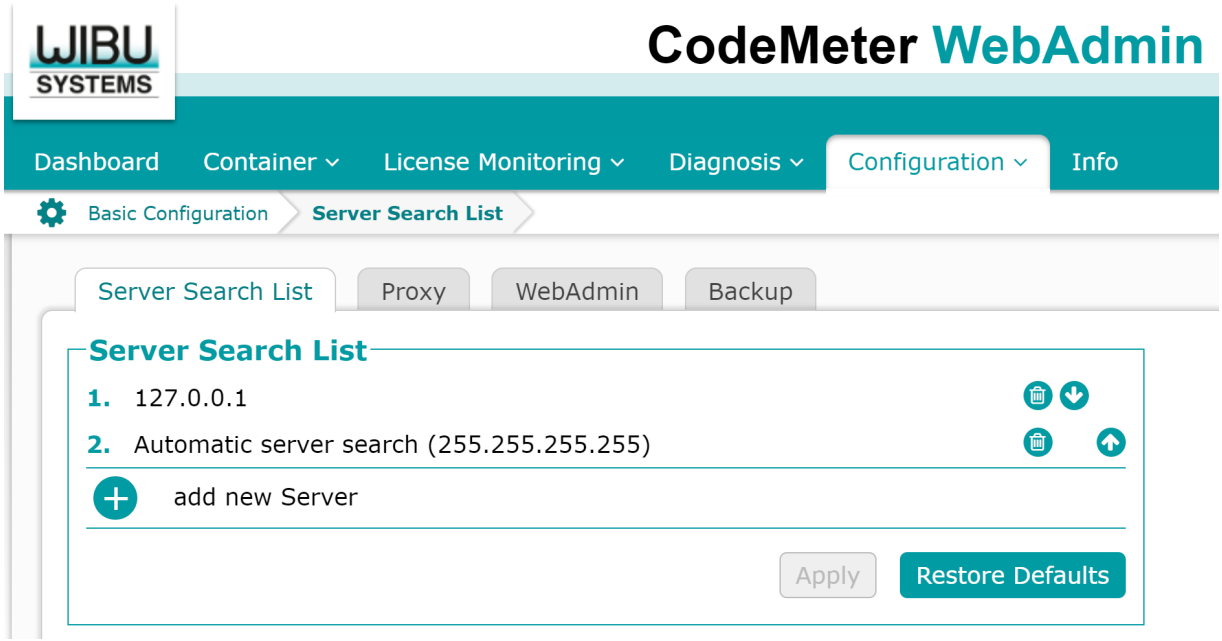


The sample above forces the usage of a v3 license giving precedence to a network license over a local license.

3.4.4 License Locations

Licenses can be either on the network, on a WIBU dongle or in a software container. The default order of priority to acquire a license is **network first** and then any **local** dongle or software container.

If you need to explicitly disable any network license location, you need to configure the loopback device `127.0.0.1` as your only source in the WIBU Server Search List.



Information: This setting might impact other locally installed software packages like the **Viz Engine** that might require a network license.

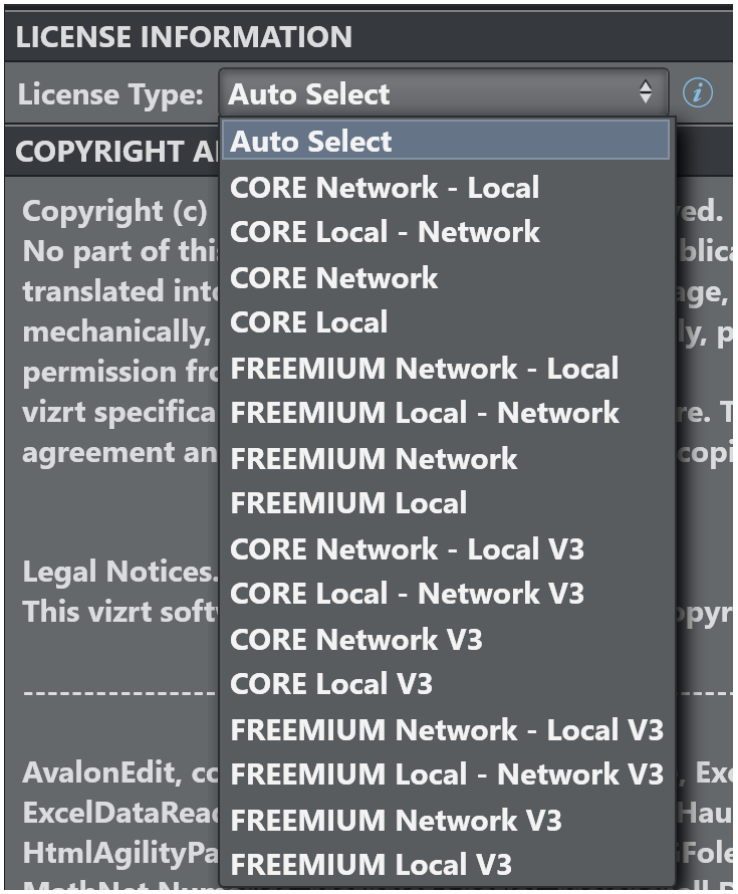
3.4.5 License Configuration

The type of license to acquire can be configured in the configuration panel under **LICENSE INFO**.

Copyright © 2025 Vizrt



Use the **License Type** dropdown to select the license type and location preference.



- **Auto Select:** Selects the most powerful license that can be acquired (thus first CORE and the FREEMIUM) automatically, searching first for a license on the network and then locally.
- **CORE Network - Local:** Forces the allocation of a a CORE license, searching first on the network and then locally.
- **CORE Local - Network:** Forces the allocation of a a CORE license, searching first locally and then on the network.
- **CORE Network:** Forces the allocation of a a CORE license over the network.
- **CORE Local:** Forces the allocation of a a CORE license locally.
- **FREEMIUM Network - Local:** Forces the allocation of a a FREEMIUM license, searching first on the network and then locally.
- **FREEMIUM Local - Network:** Forces the allocation of a a FREEMIUM license, searching first locally and then on the network.
- **FREEMIUM Network:** Forces the allocation of a a FREEMIUM license over the network.
- **FREEMIUM Local:** Forces the allocation of a a FREEMIUM license locally.

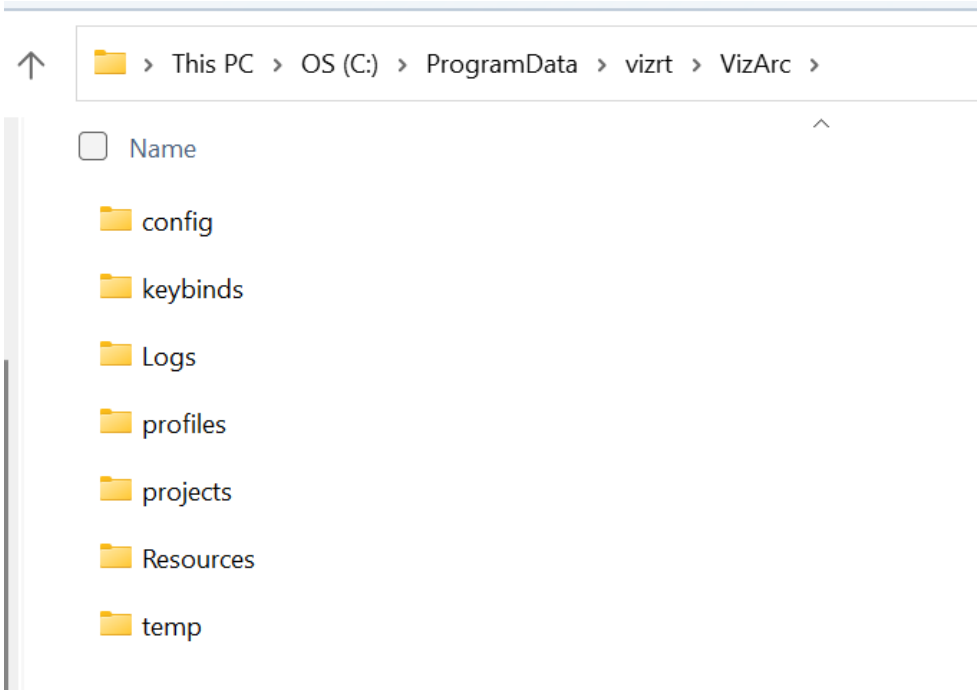
All the above licenses options exist also as **V3** variation, for the case the Version 3 license needs to be used.

Please contact your local Vizrt sales representative for any additional licensing information.

Copyright © 2025 Vizrt

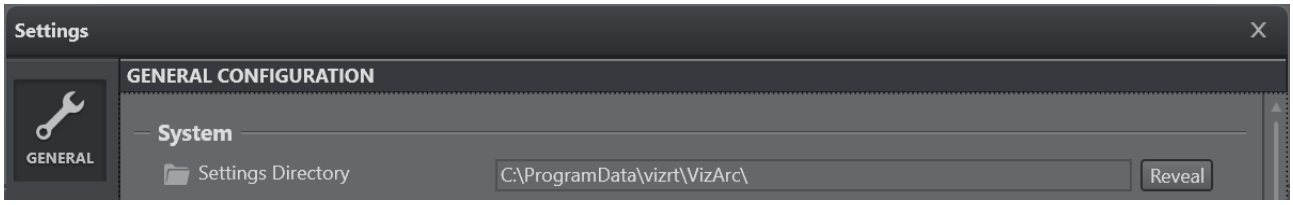
3.5 Files and Folders

Files and folders that are used or modified by Viz Arc are listed below.



Depending on the logged user, related files and folders can be found under *%localappdata%\vizrt\VizArc*. When the user has administrative privileges, the location is *c:\ProgramData\vizrt\VizArc*.

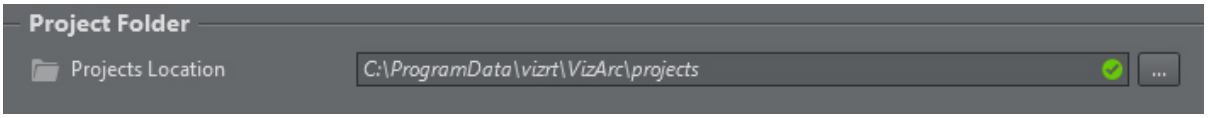
The read-only folder can be seen in the General Configuration Settings, based on the logged user launching Viz Arc.



The first time a non-administrative user launches Viz Arc, the config and profile settings are copied from *c:\ProgramData\vizrt\VizArc*, to the local application data folder. This way, an administrator can create a global default configuration for all users logging onto the system.

- **config:** Contains *config.json* file with all default and custom Viz Arc configurations.
- **keybinds:** Contains *keybinds.json* file used to determine system keyboard shortcuts.
- **Logs:** Contains a text log file containing every log name, date, log level and message since the first launch of the application.
- **profiles:** Contains *profiles.json* file where configured profiles and corresponding *Channels* and *Engines* are stored.
- **projects:** Contains a sub-folder and *.json* file for each saved Viz Arc project. You can change project folders in the **Settings > Application Settings** panel.

Copyright © 2025 Vizrt

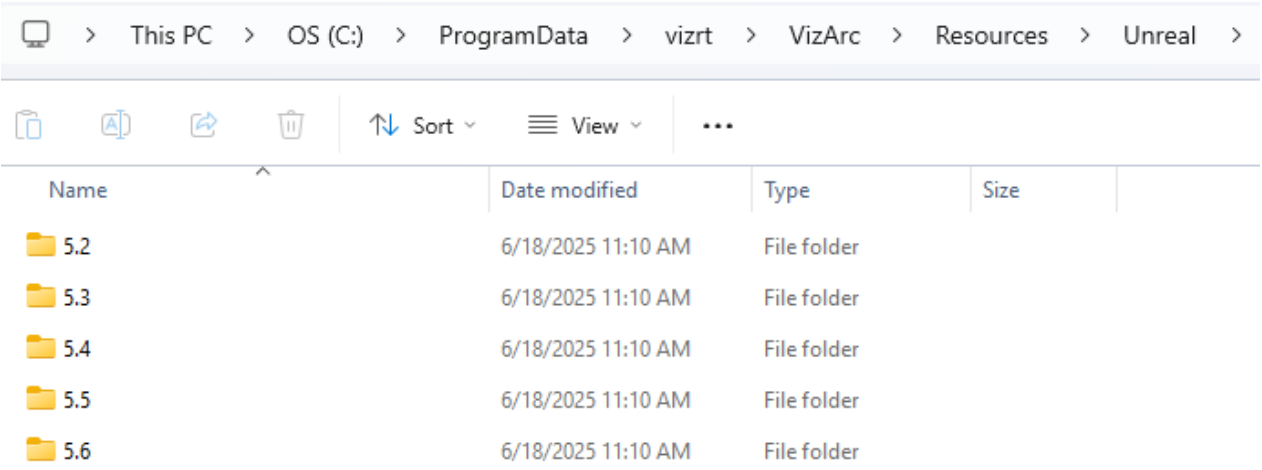


- **Resources:** Contains useful resources (for example, plug-ins for external devices, Unreal plug-ins and Viz archives).
- **temp:** Contains temporary files that are automatically created and deleted in case of a software crash.

Copyright © 2025 Vizrt

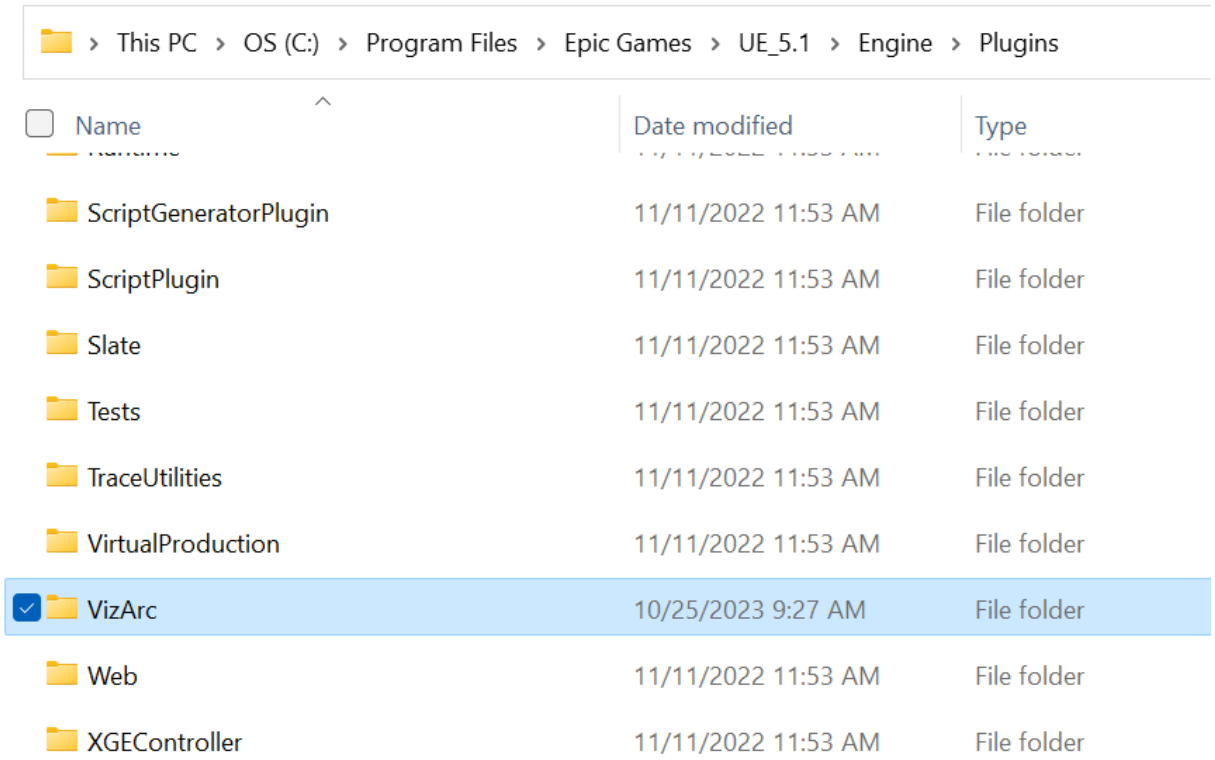
3.6 Unreal Plug-in

The Unreal plug-ins can be found in the program-data folder (typically in *C:\ProgramData\Vizrt\VizArc*) under *Resources\Unreal*.



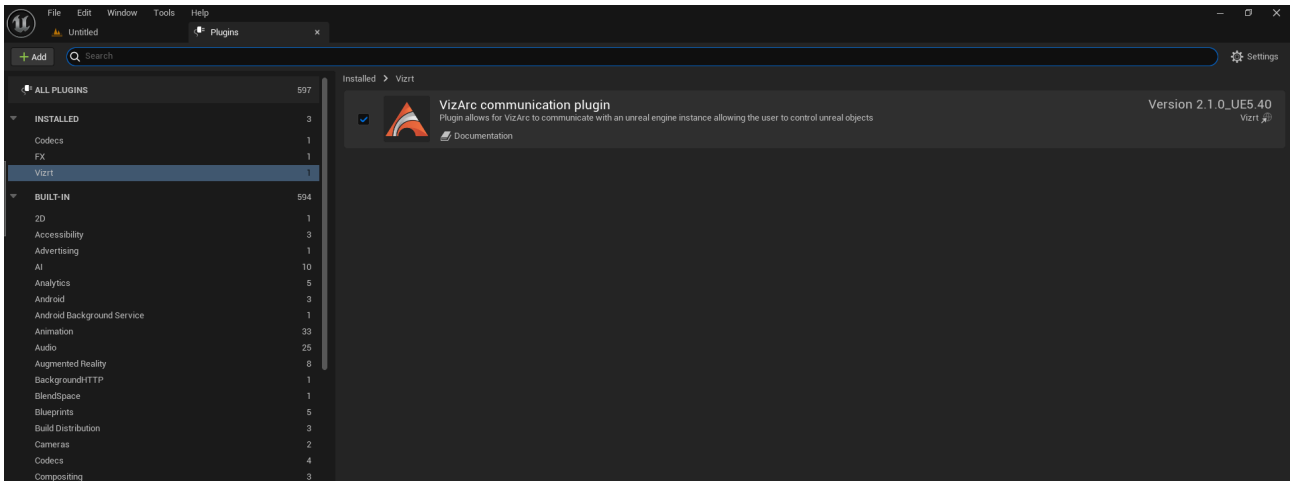
The plug-ins are usually installed automatically. However, in case an Unreal Installation was not detected or not present during installation, the plug-ins can be copied manually from the above resource folder.

Navigate into the subfolder matching your Unreal version. From within the respective subfolder, copy the entire *VizArc* folder into the Unreal installation folder (for example, *C:\Program Files\Epic Games\UE_5.3*) under *Engine\Plugins* or under *Engine\Plugins\Runtime*.



Copyright © 2025 Vizrt

After copying the plug-in, restart Unreal Engine and ensure the plug-in is activated (it is active by default). To verify, open Unreal Engine and go to **Edit** > **Plugins**. You should find the Viz Arc plug-in by searching for **VizArc**.



The communication port that is used by this plug-in in order to receive commands from Viz Arc is by default `5647`. It can be changed by changing the registry key in **UEPluginPort** in *HKEY_LOCAL_MACHINE\SOFTWARE\vizrt\Viz Arc*.

See Also

- Engine Service

Copyright © 2025 Vizrt

3.7 Engine Service

Engine Service is a scheduled task that needs to be installed on each Unreal Engine machine that is controlled by Viz Arc. Engine Service detects installed Unreal Engine versions, and helps launch a project and select a map within the project. It is not mandatory to install **Viz Arc** or the **Viz Arc Web Service**. The Unreal Loader needs to be installed on any machine running an Unreal Engine that is ought to be controlled by Viz Arc, which might run on a different machine.

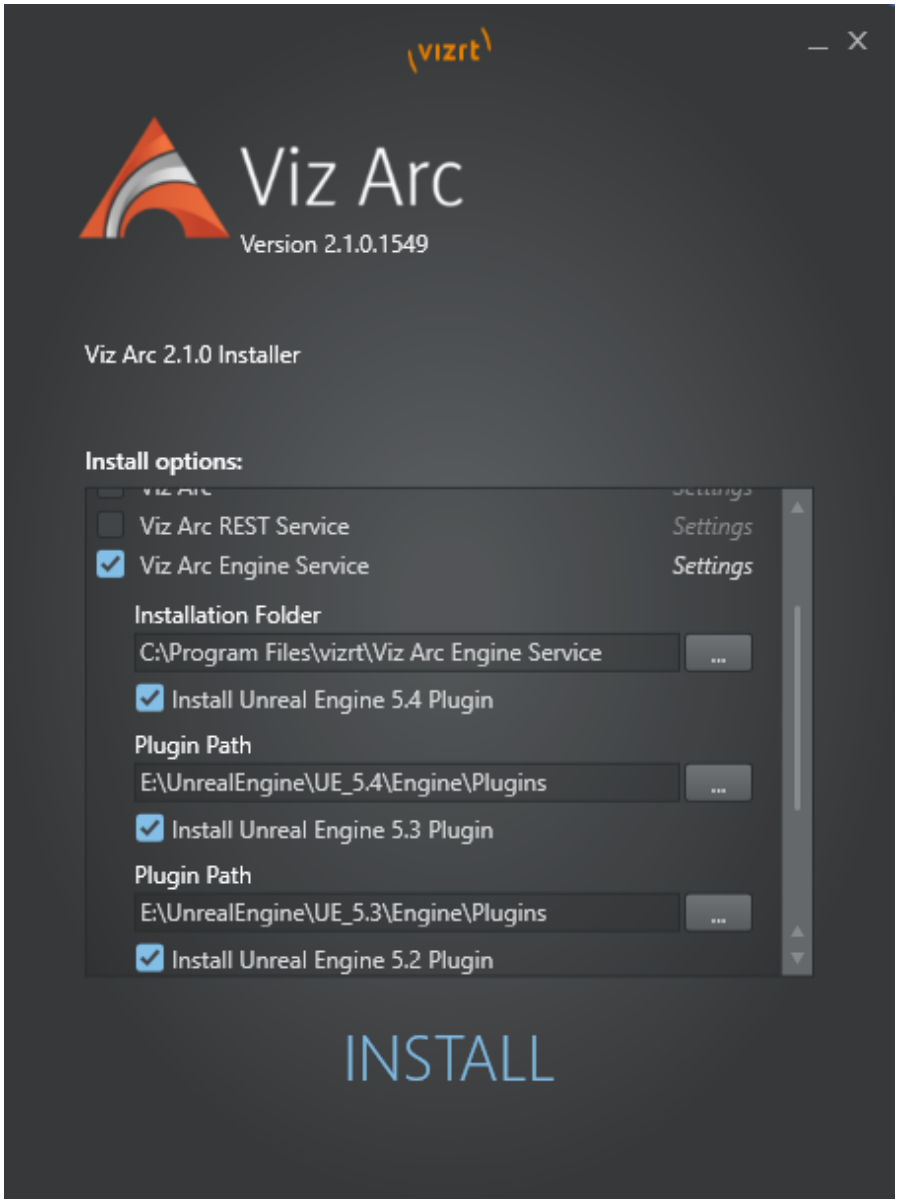
3.7.1 Prerequisites

Make sure the **MS Visual C++ 2015-2022 Redist (x64)**, **Windows Desktop runtime** and **Asp.Net Core runtime** are installed on the target machine. The bundle installer silently installs them in case they are not present on the target machine.

3.7.2 Installation

Run the Viz Arc bundle installer and select the **Viz Arc Engine Service**:

Copyright © 2025 Vizrt



The installer detects any compatible **Unreal Engine** version and installs the Viz Arc plug-in to the respective Unreal installations.

The registry key used is **InstalledDirectory** and can be found in the registry under:

- *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine\5.2*
- *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine\5.3*
- *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine\5.4*
- *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine\5.5*
- *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine\5.6*

Copyright © 2025 Vizrt



The plug-in gets installed under the subfolder **Engine\Plugins\VizArc** of the respective Unreal Engine installation (for example, *C:\Program Files\EpicGames\Unreal Engine\5.3\Engine\Plugins\VizArc*).

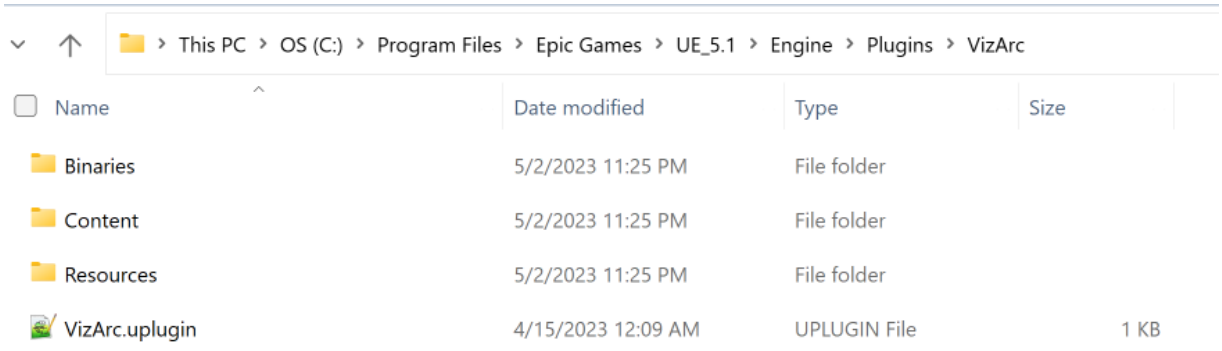
Unreal Registry Entries: On some machines, the registry entries might not be present although the Unreal Engine is installed correctly. This usually occurs when the Unreal Engine installation has been executed without administrative privileges. The registry entries can either be added manually or the plug-in can be installed manually (see below).

3.7.3 Manual Installation

The plug-ins can be copied manually to the Unreal Engine plug-in folder. After installation of the Unreal Loader, the plug-ins can be found under the folder *C:\ProgramData\vizrt\VizArc\Resources\unreal*.

Select the version to install and copy the entire **VizArc** folder to the Unreal plug-in destination folder. For example, copy *C:\ProgramData\vizrt\VizArc\Resources\unreal\5.3\VizArc* to *C:\Program Files\EpicGames\UE_5.4\Engine\Plugins\VizArc*.

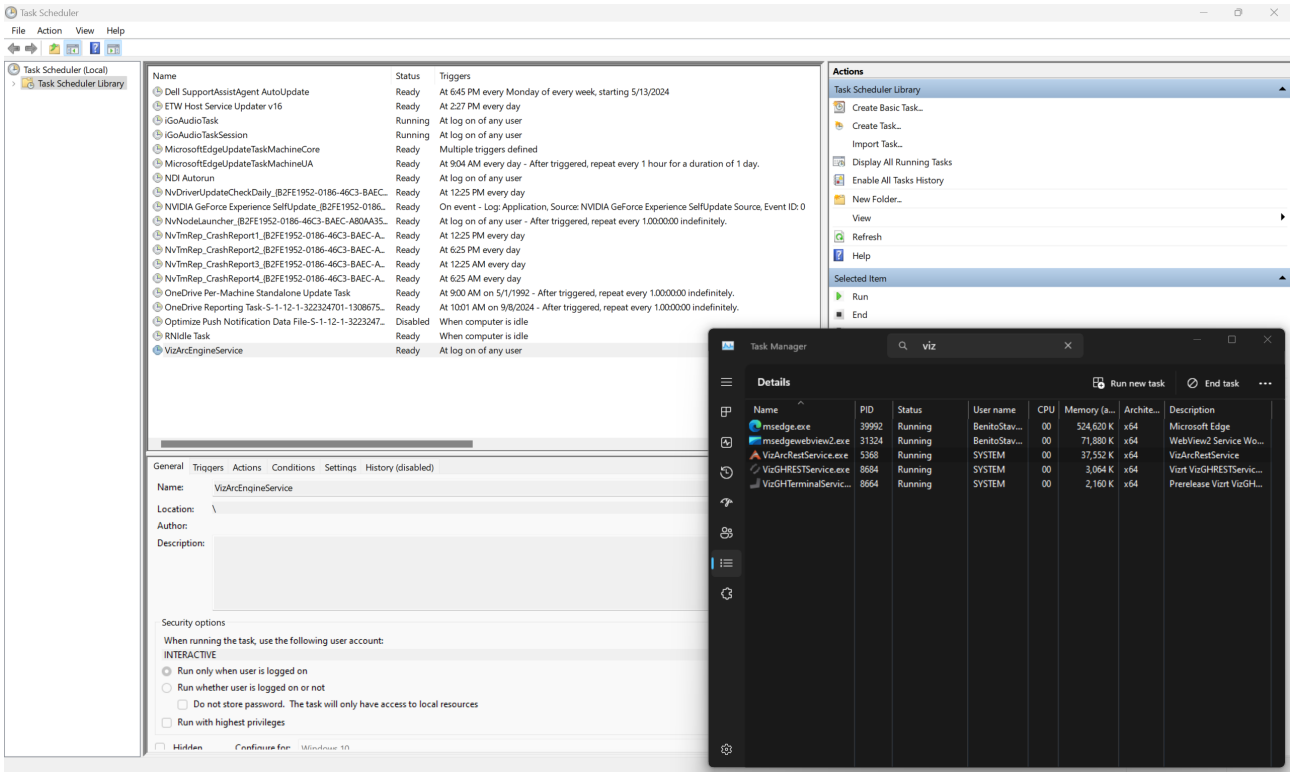
Warning: Different versions are not compatible.



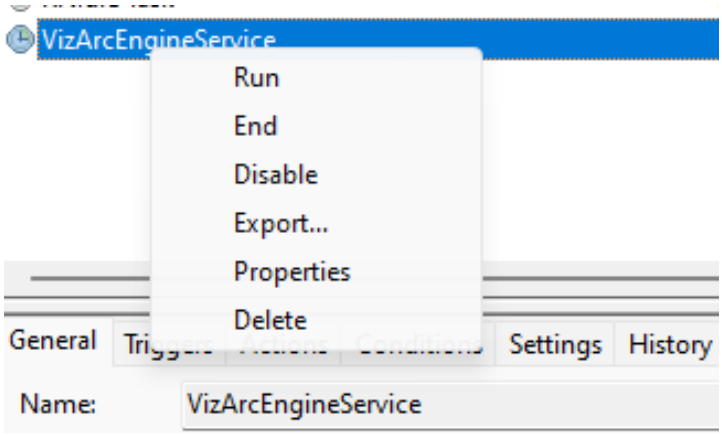
3.7.4 Starting Engine Service

After successful installation the Unreal Loader starts automatically and whenever a user logs on to the system it starts as a scheduled task. To see the task settings execute the windows command *taskschd.msc*.

Copyright © 2025 Vizrt



Under the name *VizArcEngineService* all the settings of the scheduled task can be reviewed. Check the task manager whether the process is running. The process can be terminated and started using the context menu on the scheduled task.

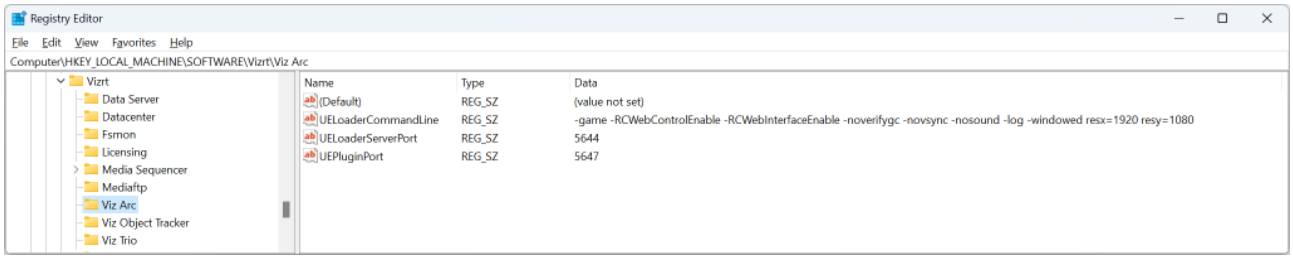


> **Note:** The service itself does not require any licenses.

3.7.5 Configuration

The first time you start the service, registry keys are generated with default values for communication ports and unreal startup parameters. The registry key can be found in *HKEY_LOCAL_MACHINE\SOFTWARE\vizrt\Viz Arc*.

Copyright © 2025 Vizrt



The relevant keys and their default values are:

- **UELoaderServerPort:** The communication port used by Viz Arc to talk to this service. Default is `5644`.
- **UELoaderCommandLine:** The default command line suffix added on each Unreal Engine start up. Default is `-game -RCWebControlEnable -RCWebInterfaceEnable -noverifygc -novsync -nosound -log -windowed resx=1920 resy=1080`.

> ⚠ **Note:** The default *resx* and *resy* are configured to be HD format. Change this accordingly if your output format is different. Add the *log* flag to launch the engine in offscreen mode with the output console only.

> ⚠ **Render off-screen:** When used together with the Viz Engine integration, it is recommended to add the `-RenderOffScreen` switch to the command line, so the rendering takes place off screen.

- **UEPluginPort:** The communication port to be used by Viz Arc to talk to the Viz Arc Unreal Engine Plugin. Default is `5647`.

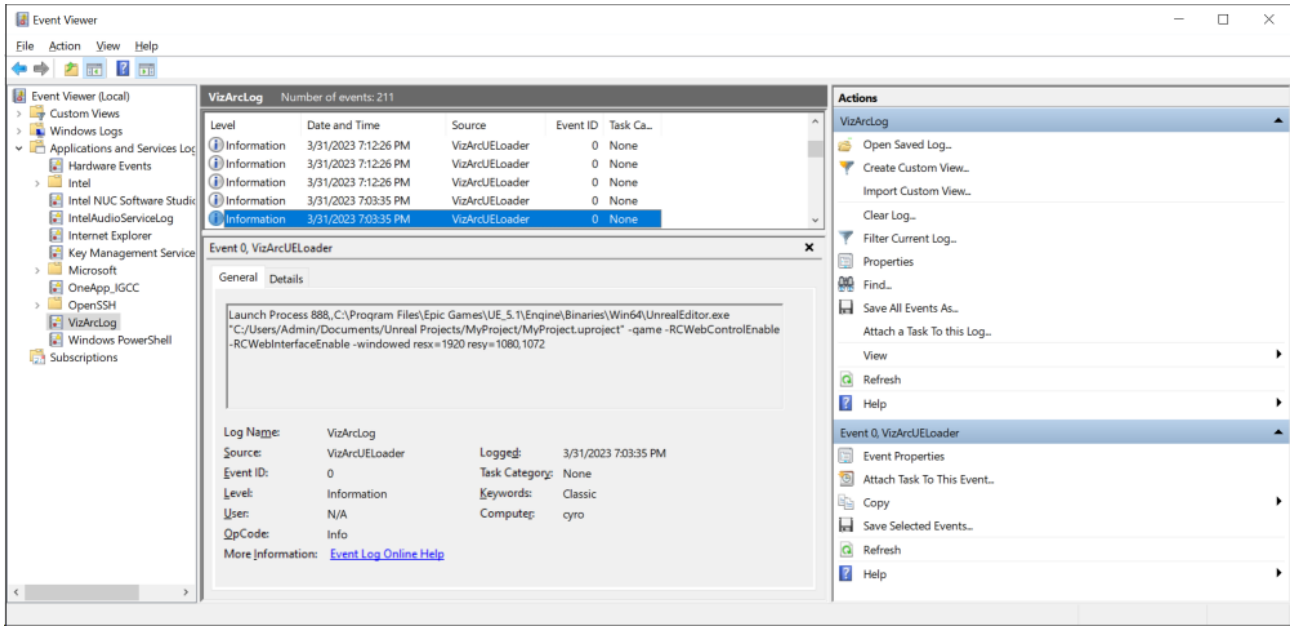
> ⚠ **Note:** The UEPluginPort is not used directly by Unreal Loader.

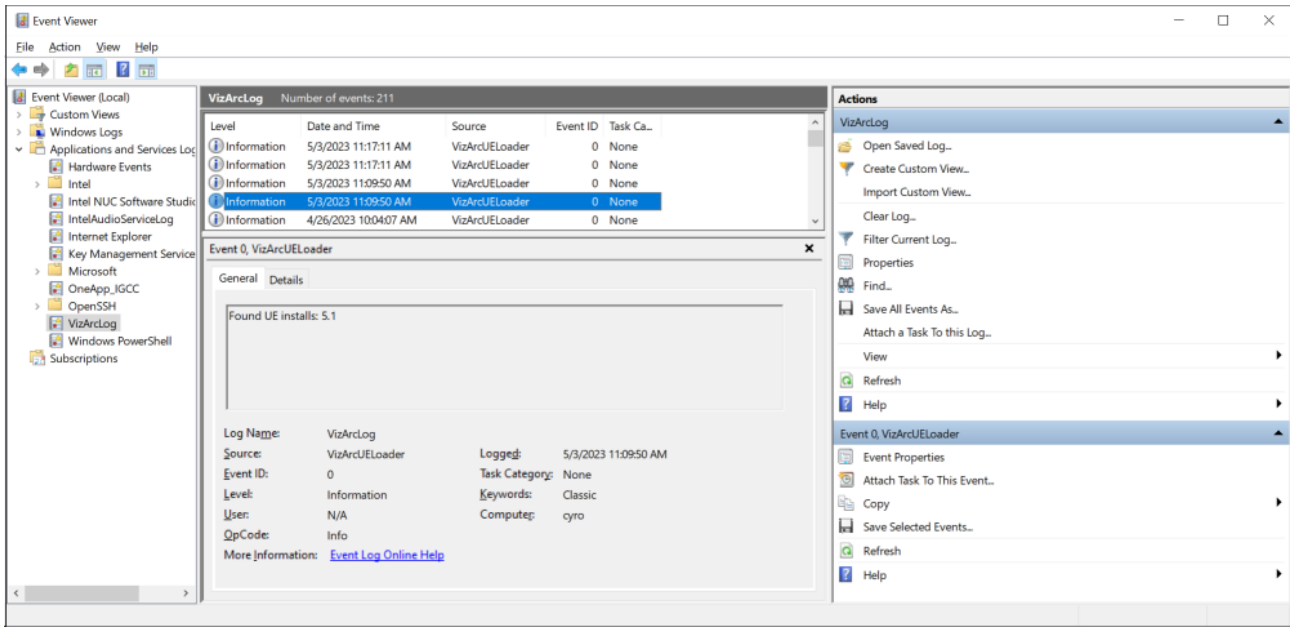
3.7.6 Troubleshooting

Logging

The Windows event viewer contains some basic logging of the service. Under **Application and Services** you can find a **VizArcLog** that contains all relevant log entries:

Copyright © 2025 Vizrt





Among other events the event viewer logs which project and map is loaded through Viz Arc and which Unreal Installations have been found after start-up. The Unreal Loader identifies the Unreal installations based on the registry keys found in *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\EpicGames\Unreal Engine*.

User

The service launches the Unreal Engine with the user that is currently logged on. Make sure a user is logged in on the machine.

Copyright © 2025 Vizrt

3.8 Configuring Graphic Hub

3.8.1 Configuring Graphic Hub for the Viz Arc Template Workflow

Note: This step is only necessary if you want to use the template based workflow or if you want to store projects on Graphic Hub and is not required for other use cases.

As there is currently no way to automatically generate and publish add-ons in Viz Graphic Hub, a few manual steps must be performed in Graphic Hub. You need to use Viz Graphic Hub Manager to connect to your Graphic Hub.

There are two ways to create the necessary **add-ons** on Graphic Hub: importing an archiving or creating them manually. This section covers the following topics:

- Configuring Graphic Hub for the Viz Arc Template Workflow
- Import an Archive Containing Add-ons
- Manually Creating Add-ons

3.8.2 Import an Archive Containing Add-ons

The archive to be imported anywhere on the Graphic Hub is located in the program-data folder (typically *C:\ProgramData\vizrt\VizArc*) under the *Resources\AddOns* folder and is called *VizArcAddOns.via*. It contains a dummy image with all necessary add-ons attached.

You can also import the add-ons through Viz Arc. When Viz Arc detects a missing add-on, e.g. on the template section, you can use the **Create Add-Ons** button to import the above archive automatically. After importing the archive you will be able to create and import templates and store projects on Graphic Hub.

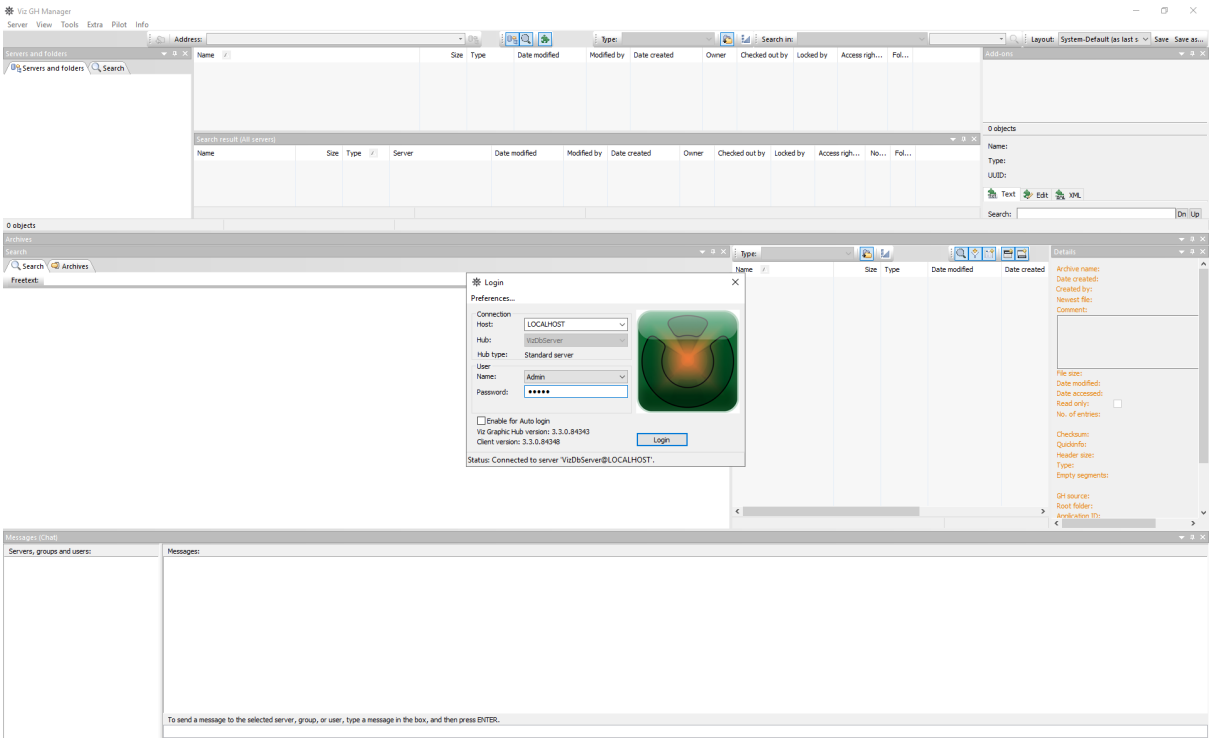
Copyright © 2025 Vizrt



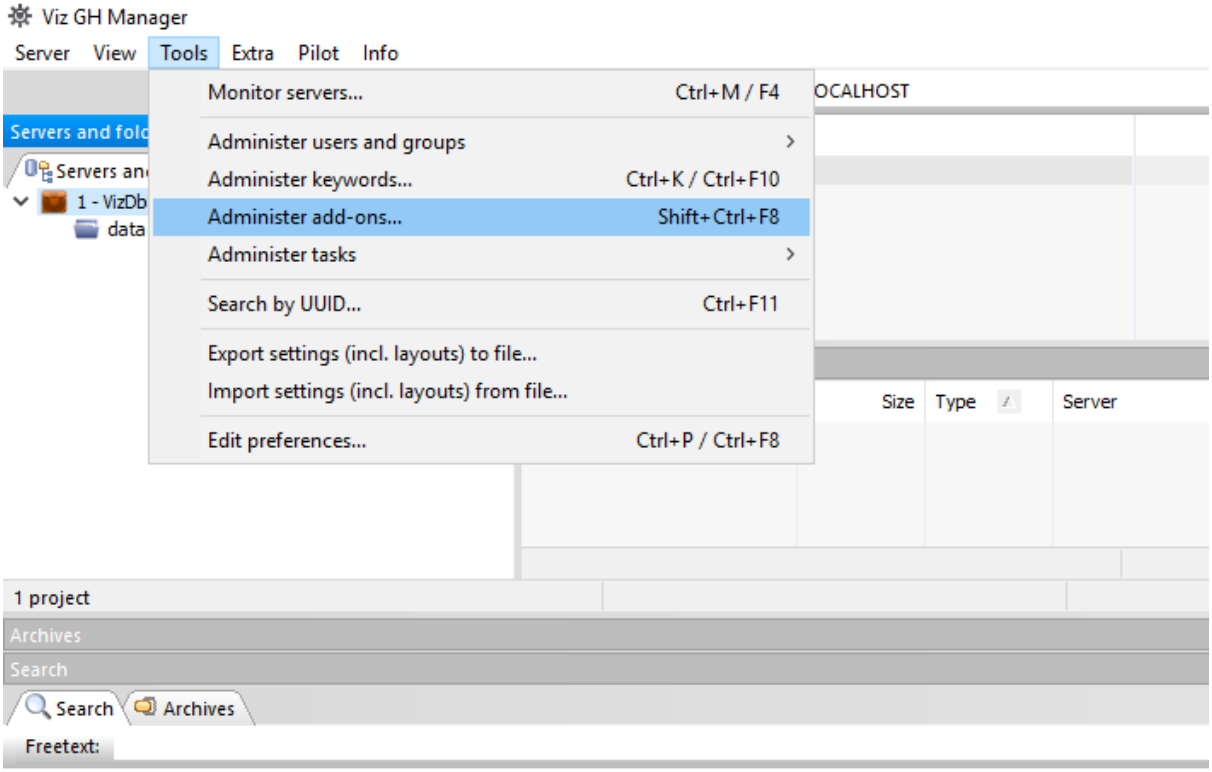
3.8.3 Manually Creating Add-ons

1. Log in to Graphic Hub using Graphic Hub Manager:

Copyright © 2025 Vizrt

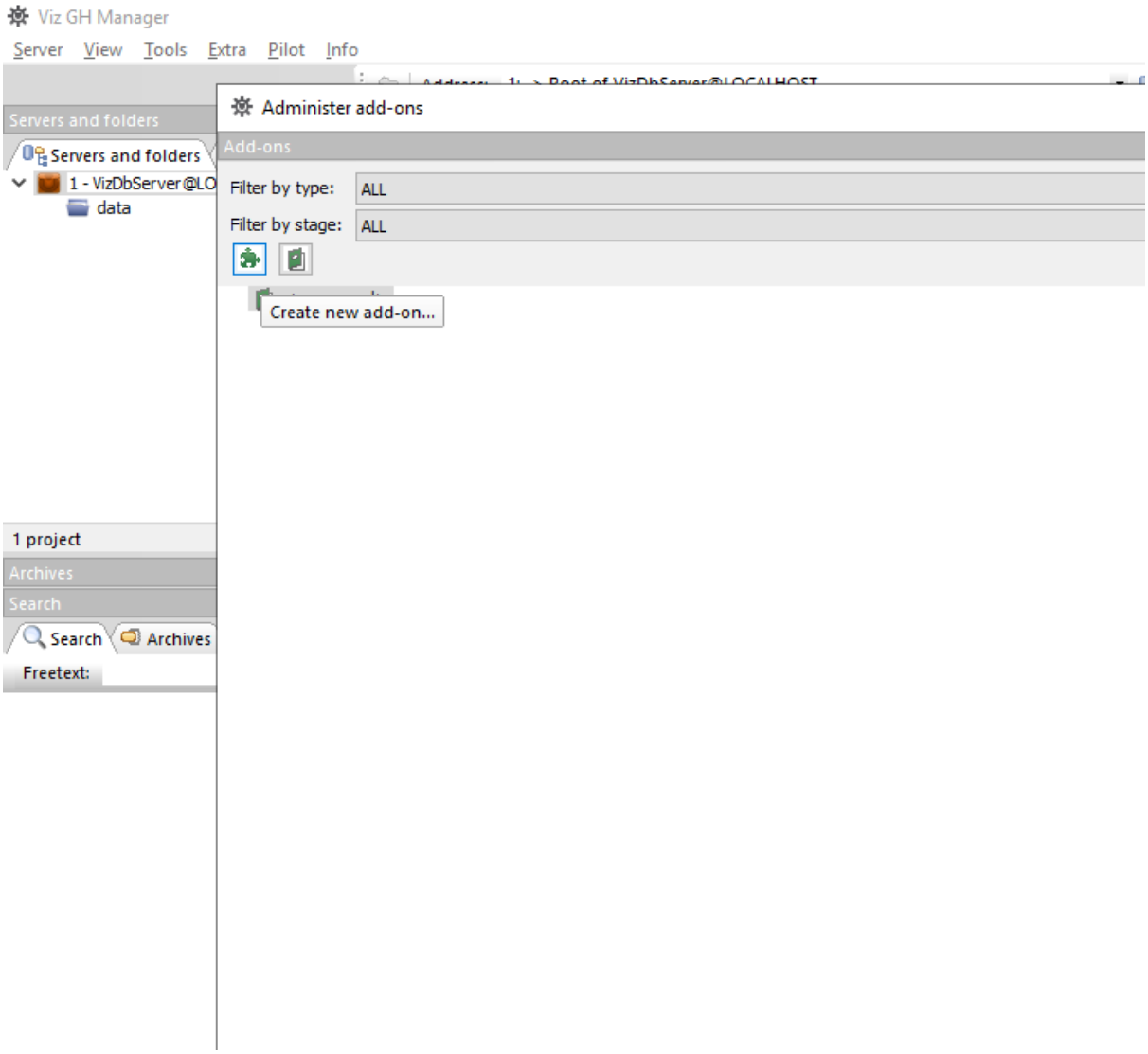


2. Go to **Tools** and select **Administer add-ons...**:

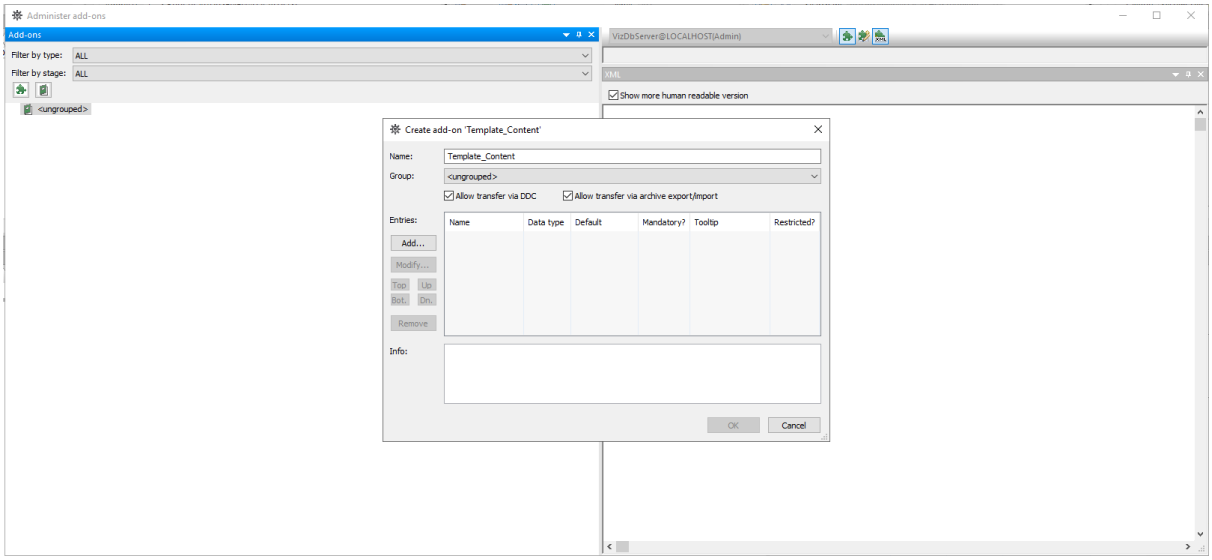


3. Create a new add-on.

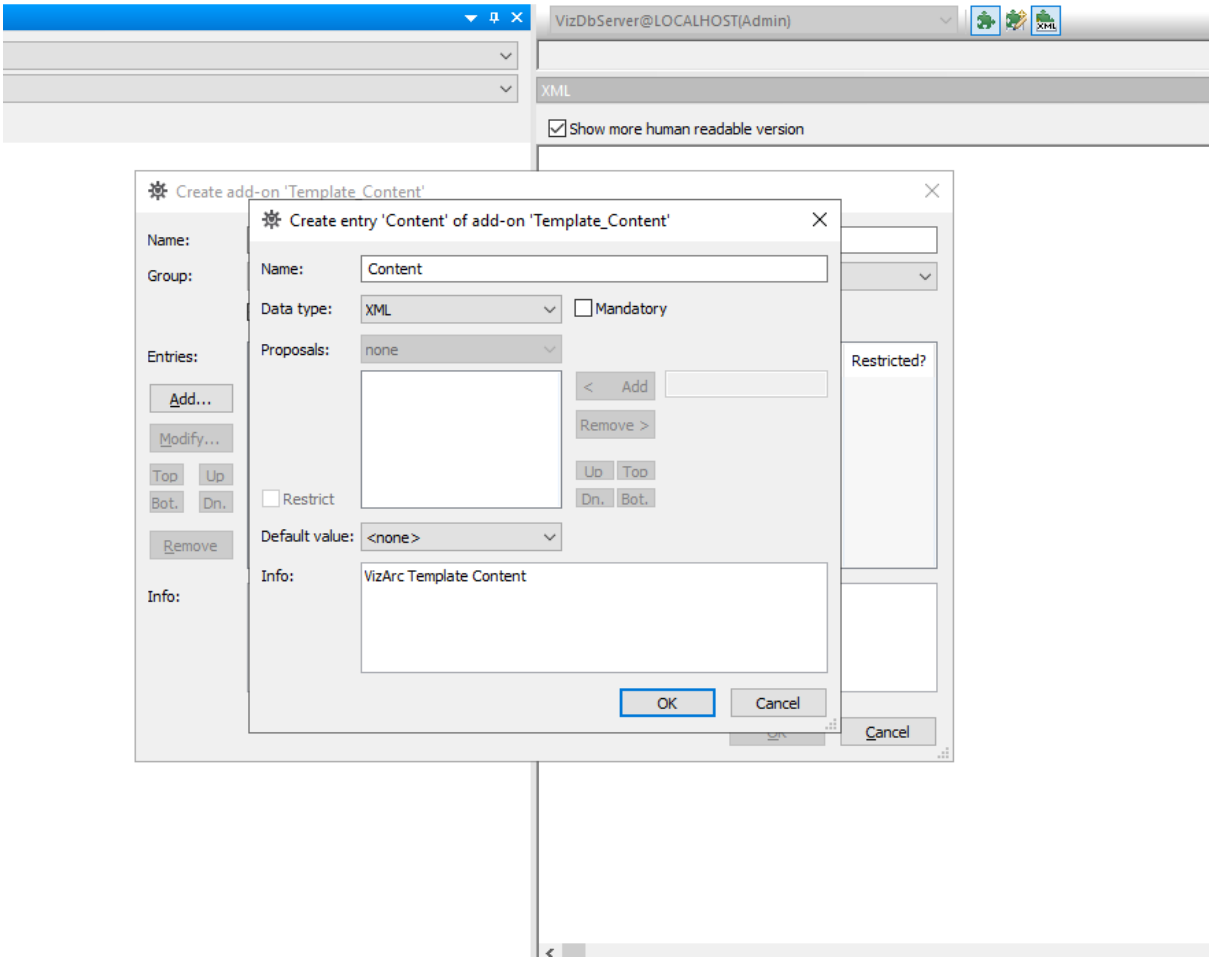
Copyright © 2025 Vizrt



4. Name the new add-on *Template_Content*. Note that the name is case-sensitive. Tick the boxes **Allow transfer via DCC** and **Allow transfer via archive export/import**.

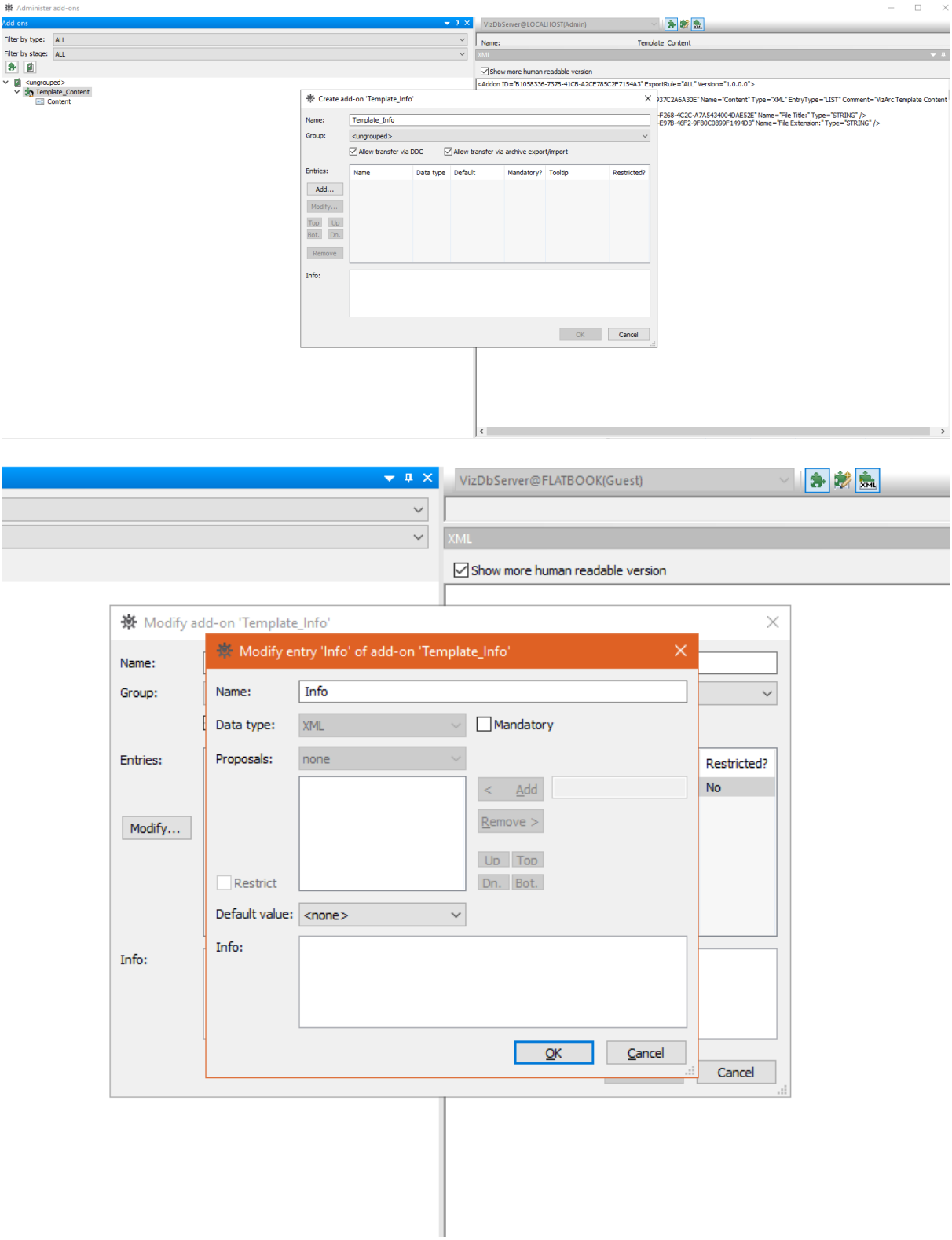


5. Click **Add...** to add an entry.
    - Call the new entry *Content*.
    - Select **XML** as data type.
    - Add something like *Viz Arc Template Content* under **Info**.



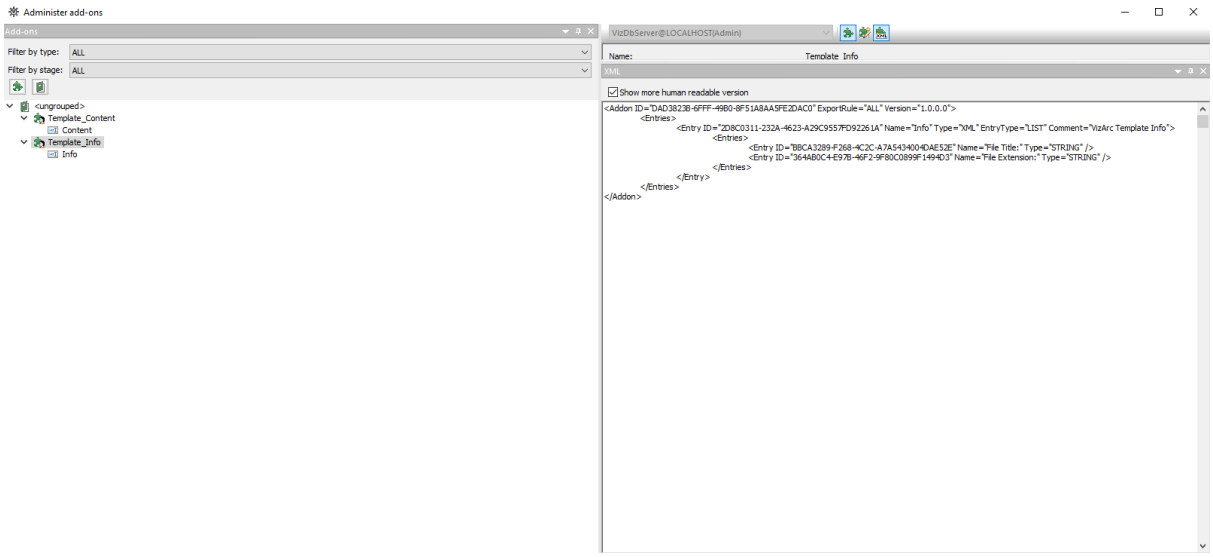
Copyright © 2025 Vizrt

6. Return to step 3 above and add another add-on:
    - Name the add-on *Template_Info*. Bear in mind that the name is case-sensitive.



- Add an entry under **Info**, something like *Viz Arc Template Info*. The data type must be **XML**.

Copyright © 2025 Vizrt

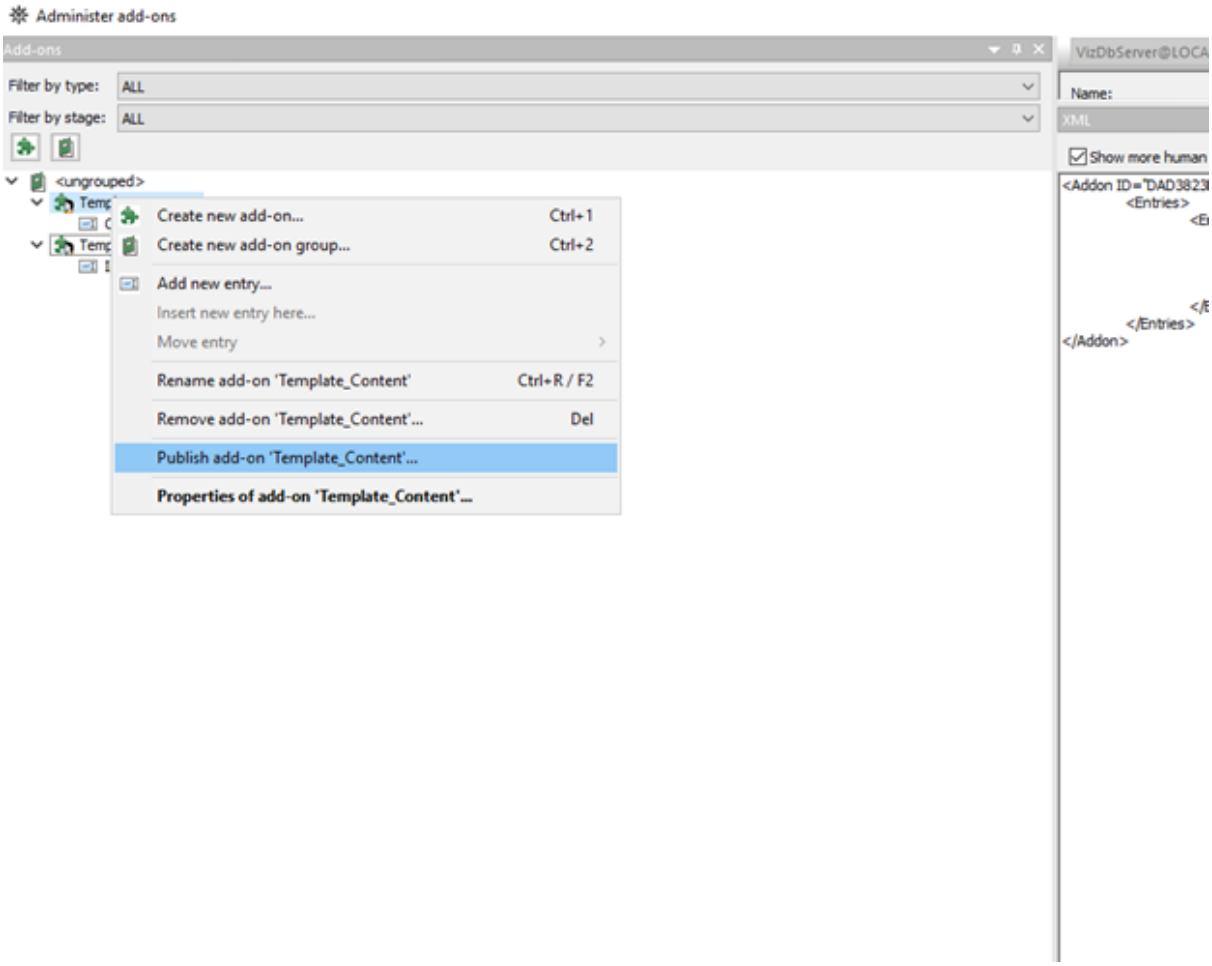


Your add-ons should now look like those in the screenshot above.

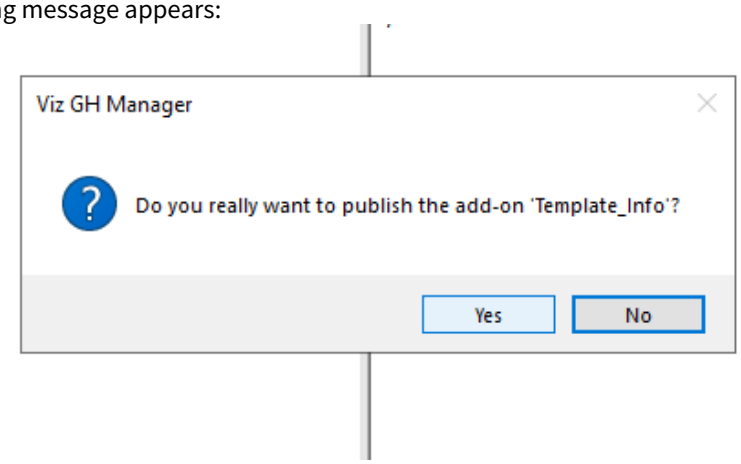
Note: You can add the add-ons to a group if you wish. Viz Arc finds them regardless of which group they belong to.

8. Publish by right-clicking the two add-ons and selecting **Publish add-on ....**

Copyright © 2025 Vizrt



A warning message appears:



- Click **Yes**.
- Make sure to publish both add-ons.

Once you have restarted Viz Arc you can use the **Design** tab to create your own templates.

Copyright © 2025 Vizrt

3.9 Script Debugging

To get useful information on the Viz Arc scripting console when errors occur, it's necessary to install a debugger on the target system.

3.9.1 How to Enable Viz Arc Script Debugger on Windows 7/8/8.1/10

Issue Details

I would like to debug my script (main or template script) in Viz Arc, but I don't get the line of the occurrence on the console output when the script runs into errors. What can I do?

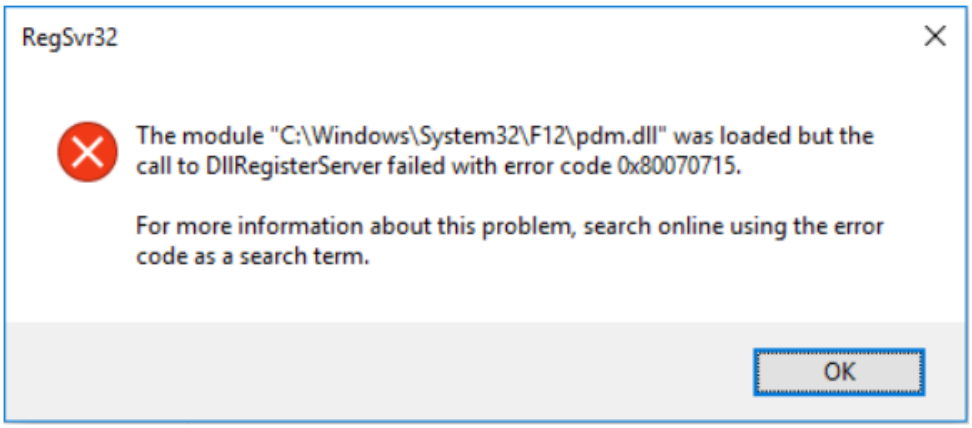
Solution

Viz Arc

The registration process of the *pdm.dll* differs depending on the version of the OS. To register the *.dll*, run cmd in admin (elevated) mode and type the command below that corresponds to your software configuration:

- **Windows 7/8/8.1:** `Regsvr32.exe "C:\Program Files\Internet Explorer\pdm.dll"`
- **Windows 10 (up to version 1511):** `Regsvr32.exe "C:\Windows\System32\F12\pdm.dll"`

With Windows 10 version 1607 and above, *pdm.dll* can no longer be registered manually, but must be automatically registered by a Microsoft product. If attempting to register manually, *RegSvr32.exe* throws error 0x80070715.



One way to automatically register the *.dll* is to install Microsoft's Remote Tools for Visual Studio. Download the x64 version. This free tool will install and register the PDM. After installation, Viz Arc will automatically recognize the registered *.dll*.

Additional Information

Viz Arc Script is based on ClearScript which relies on the Microsoft Script Debugger that was released in 1997. This tool could be used to debug Visual Basic scripts (VBScript) or Java scripts (JScript) and was a complement to products such as Internet Explorer 4.0 and Internet Information Server 4.0. Until recently, it was a tool that Microsoft provided as a separate download on their homepage. It installed a shared library called the Process

Copyright © 2025 Vizrt

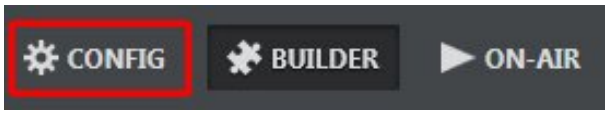
Debug Manager (PDM), which Viz Arc/ClearScript uses to debug its own VBScripts/JavaScript.

With Windows 7/8/8.1/10, users do not need to install this file separately, provided they have either MS Office or Visual Studio installed. It also installs with Internet Explorer 8 and above but is not registered (see the steps for registering the *pdm.dll* above). As the PDM library is already available on these systems, you only need to make sure the active *.dll* is registered in order for Viz Arc to recognize it.

Copyright © 2025 Vizrt

4 Configuration

To access the configuration panel, press the **CONFIG** icon on the top right of the main window or go to the main menu and select **System > Configuration**.



This section covers the following topics:

- General Configuration
- General Settings
- Profiles
- Keyboard Shortcuts
- Video Settings
- GPI Setup
- Timecode Setup
- Clip Setup
- Info
- Tracking
- Vizrt Systems
- Third Party Systems
- Loupedeck Integration
- Stream Deck Integration
- Preview Studio Configuration

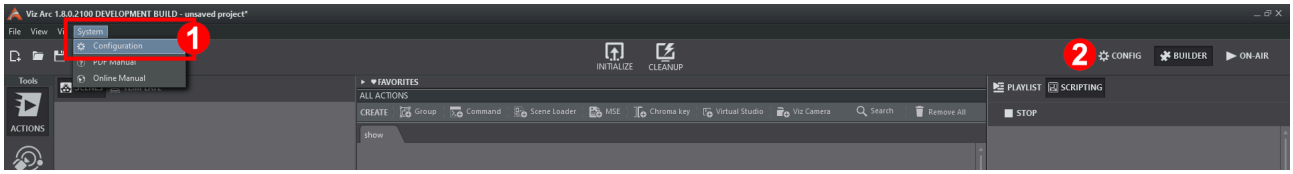
Copyright © 2025 Vizrt

4.1 General Configuration

General Configuration is the main configuration window. Here the user can set parameters that are *required* for Viz Arc to function correctly with systems like Viz Engine, Graphic Hub and Tracking Hub.

To reach the Viz Arc **Configuration** there are two ways, as shown in the picture below:

1. Click on the menu bar item **System > Configuration**.
2. Click on the ***CONFIG*** button on the main window.

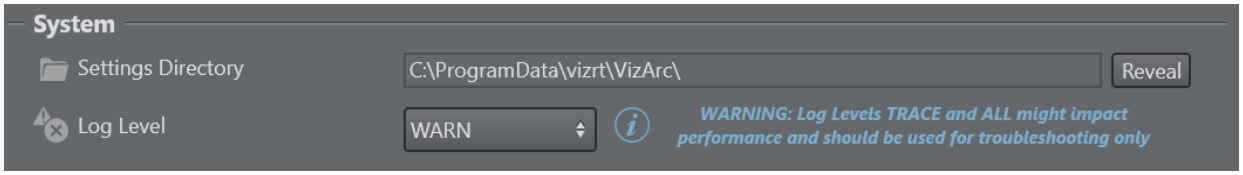


4.2 General Settings

Set application level preferences using the menu below:

4.2.1 System

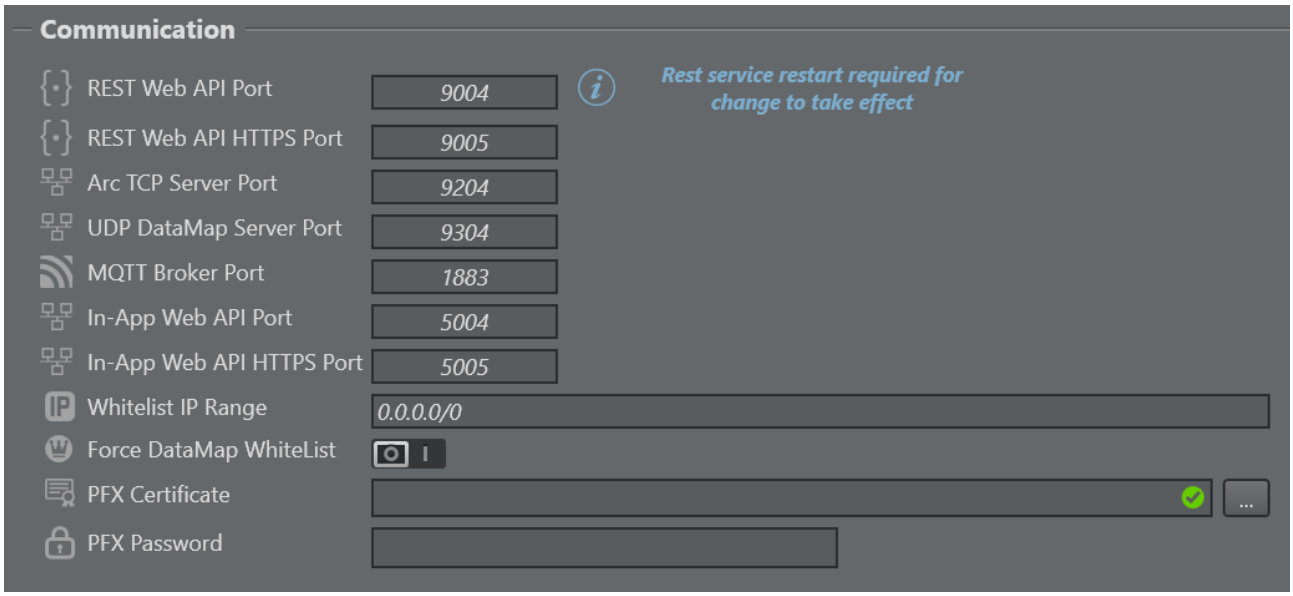
The read-only **Settings Directory** indicates where Viz Arc stores configuration and log files. For users with administrative provileges this is *c:\ProgramData\vizrt\VizArc* while normal users will find their configuration files and logs in *%localappdata%\vizrt\VizArc*. The **Reveal** button opens the respective directory location in a windows file explorer.



Set the **Log Level** using the System menu:

- **ALL:** Shows all level logs (below).
- **TRACE:** Shows traced code; useful for identifying errors in a function if there is a software crash. Logs all commands sent to the Engine in a separate log file.
- **DEBUG:** Shows specific technical information.
- **INFO:** Shows the main information in a log.
- **WARN:** Shows issues that may potentially cause application anomalies.
- **ERROR:** Shows any error that is fatal to the operation.
- **FATAL:** Shows only errors that force a shutdown of the application to prevent data loss.
- **OFF:** Switches logs `Off`.

4.2.2 Communication

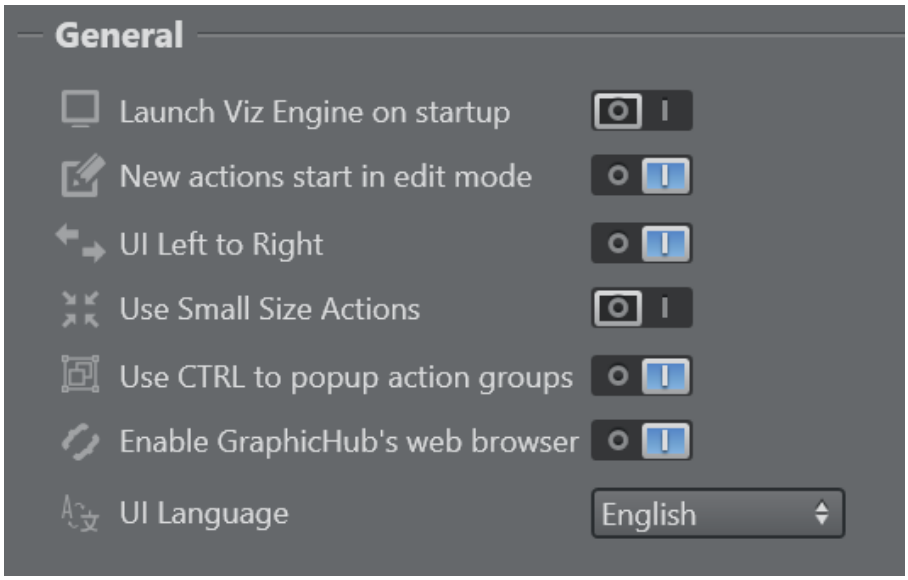


Copyright © 2025 Vizrt

- **REST Web API Port:** The REST port used for listening to incoming requests by the Viz Arc REST server.
- **REST Web API HTTPS Port:** Same as above, but for the HTTPS clients. Requires the setup of the PFX certificate parameter (below).
- **Arc TCP Server Port:** The TCP Server port to be used for incoming TCP Connections.
- **UDP DataMap Server Port:** The UDP Server port to be used for incoming UDP packets for DataMap updates.
- **MQTT Broker Port:** The internal MQTT Broker port to be used. Set to `0` to disable the Broker.
- **In-App Web API Port:** The port to be used for the internal web server.
- **In-App Web API HTTPS Port:** The port to be used for the internal web server.
- **Whitelist IP Range:** Specifies a semi-colon (**;**) separated list of IP addresses/ranges (in CIDR notation) allowed to access the web server.
- **PFX Certificate:** The path to the PFX certificate required to enable HTTPS for the REST service.
- **PFX Password:** The password of the PFX certificate.

> ⚠ **Note:** All settings take effect upon restart of the REST service or Viz Arc.

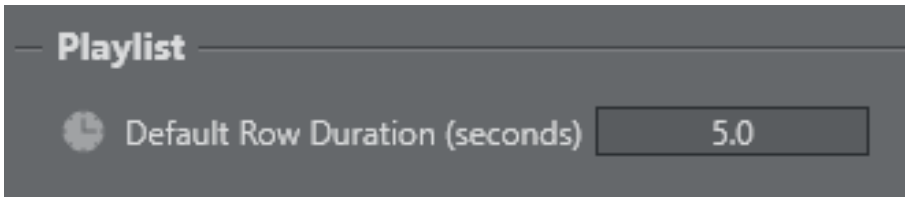
4.2.3 General



- **Launch Viz Engine on startup:** Launches Viz Engine on Viz Arc application startup, when enabled.
- **New actions start in edit mode:** Starts new actions in edit mode when created when set to `On`.
- **UI Left to Right:** Switches the UI to right to left mode (experimental) when set to `Off`.
- **Use Small Size Actions:** Uses smaller buttons for Actions when set to `On`.
- **Use CTRL to popup action groups:** Hold **CTRL** while left clicking on the **Expand** button of a Group Action to expand the group temporarily as a popup. Disable this option to make this the default behavior without holding the **CTRL** button.
- **Enable Graphic Hub's web browser:** Uses web view based Graphic Hub browser for scene browsing.
- **UI Language:** Determines the language to be used for the UI (feature under development).

Copyright © 2025 Vizrt

4.2.4 Playlist



- **Default Row Duration:** The default duration of a playlist row on creation.

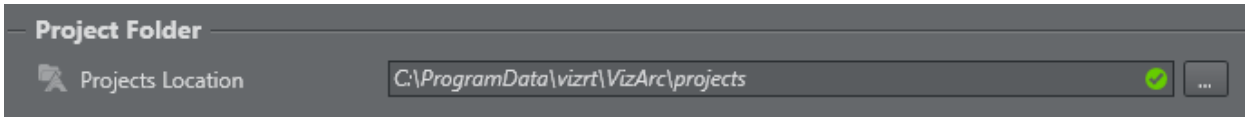
4.2.5 Scripting



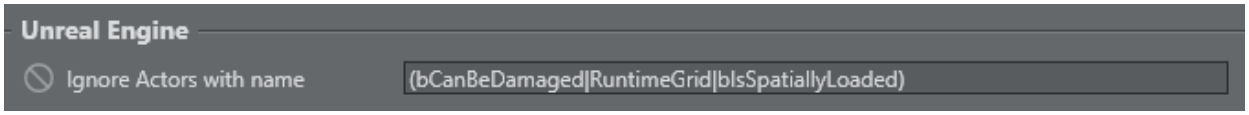
- **Compile scripts on project load:** Forces any custom script included in the project to be compiled when opening a project.
- **Main Script Debug Port:** The debug port to connect an external debugger for the main script.
- **Editor Script Debug Port:** The debug port to connect an external debugger for the script in the template designer.
- **JS Modules Search Path:** Sets the JavaScript module path. Use a semi-colon (**;**) to separate multiple paths.

4.2.6 Project Folder

This menu contains the path where projects are saved. You can change the Viz Arc project folder at any time:



4.2.7 Unreal Engine



- **Ignore Actors with name:** A regular expression (in C# syntax) that excludes Unreal Actors when parsed from a BluePrint Actor (for example, the BluePrint Action or in a Unreal Template). Leave this field empty if all Actors should be considered. Another sample could be *(bCanBeDamaged|RuntimeGrid|bIsSpatiallyLoaded).* This regular expression would exclude all actors with the exact name *bCanBeDamaged* or *RuntimeGrid* or *bIsSpatiallyLoaded*. Switch the **Log Level** to at least **INFO** to see in the log which Actors got effectively ignored (for example, after creating a BluePrint action).

Copyright © 2025 Vizrt

4.2.8 Map

A key Viz Arc feature is the ability to set up augmented reality elements, tracked cameras and polygon masks in outdoor productions directly over a map.

To use Bing maps and the search feature, please create an API key, read more here.

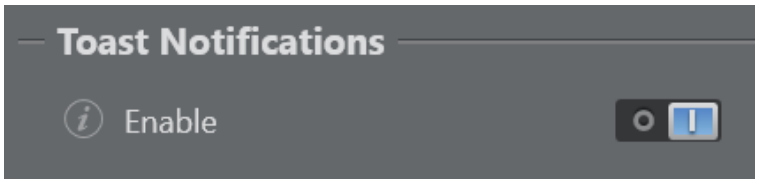
Select a **Default Provider** for map imagery using the Map menu below:



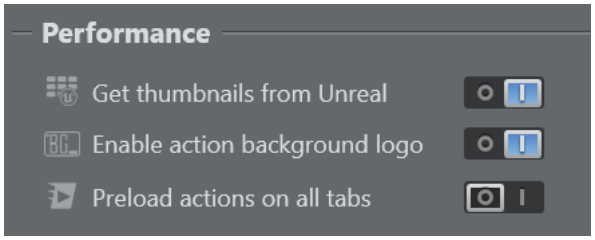
- **Default Provider:** Selects the default map provider for the map section
  - Bing Satellite, raw satellite view.
  - Bing Hybrid, satellite including boundaries/places, street names and roads as overlay.
  - Bing Just the map view showing boundaries/places, street names and roads.
- **Bing API Key:** The API Key used to download map tiles and location searches on the set view. Use the refresh button to check the entered API Key. A restart of the application is required to use the new API key.

4.2.9 Toast Notifications

Toast notifications show notifications on the bottom right of the window to show actions being triggered bz external triggers such as the web API. Thez can be enabled or disabled here.



4.2.10 Performance



Use this menu to select the following **Performance** options:

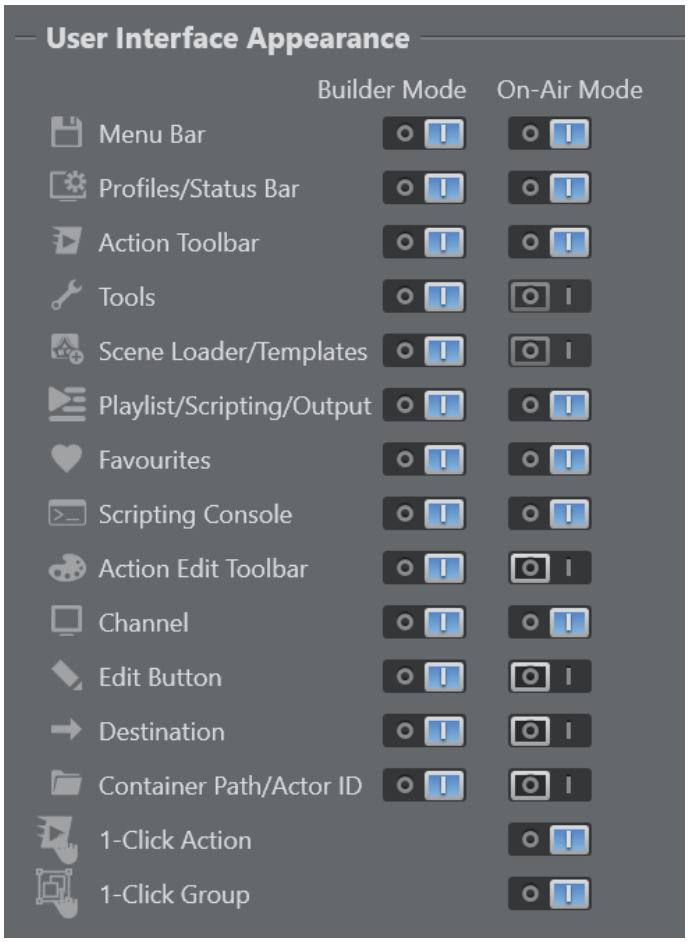
- **Get thumbnails from Unreal:** Enables the retrieval of thumbnails from Unreal Engine.
- **Enable action background logo:** Enables a watermark logo to be displayed on Actions, to clearly show the type of action.

Copyright © 2025 Vizrt

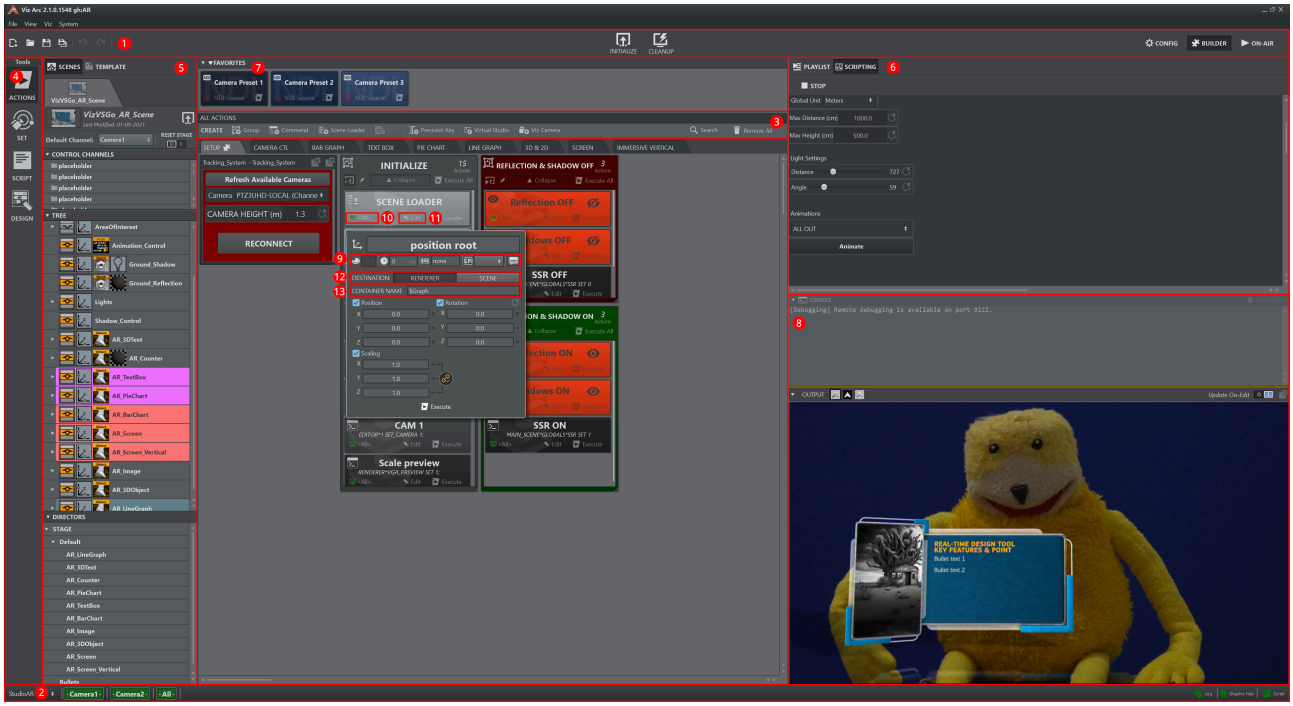
- **Preload actions on all tabs**: Enable this to preload all the action UI's on all tabs when opening a project. This will increase load times but it will eliminate loading lags when opening a different action tab for the first time. This might be convenient for large projects.

4.2.11 User Interface Appearance

Use these settings to customize the UI appearance for builder and On Air mode.

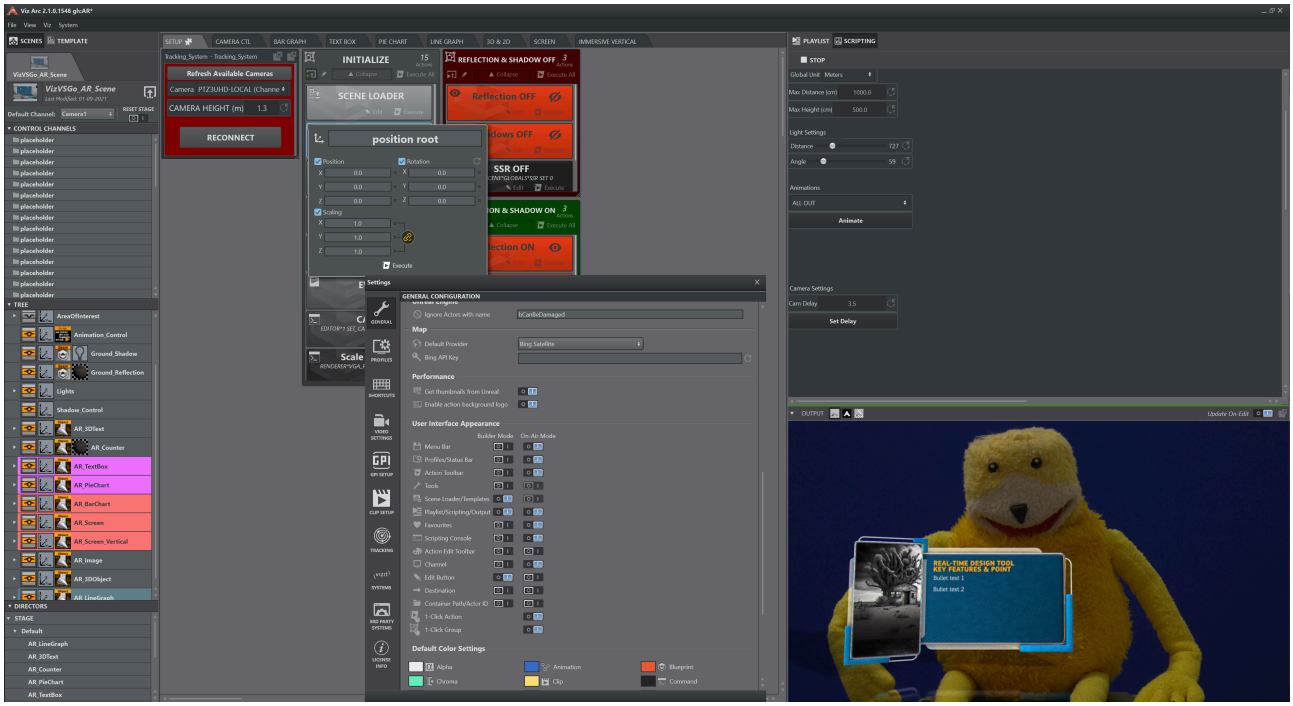


Copyright © 2025 Vizrt



1. **Menu Bar:** Toggles the menu bar containing the new, open, save, save as, undo/redo initialize and cleanup shortcuts. Most items still be reached through shortcuts or the menu.
2. **Profiles/Status Bar:** Toggles the bottom bar containing the profiles selection and the channel status information.
3. **Action Toolbar:** Toggles the toolbar containing shortcuts to create actions.
4. **Tools:** Toggles the left hand side tool bar. Tools can still be reached through the **View** menu.
5. **Scene Loader/Templates:** Toggles the left hand side of the UI containing the scene loader and the templates.
6. **Playlist/Scripting/Output:** Toggles the right hand side of the UI containing the script, playlist and output panel.
7. **Favorites:** Toggles the favorites bar.
8. **Scripting Console:** Toggles the scripting console.
9. **Action Edit Toolbar:** Toggles the action toolbar containing color selection and more when editing an action. Most properties can still be changed through the context menu.
10. **Channel:** Toggles the channel visibility on the action.
11. **Edit Button:** Toggles the action's edit button. The action can still be edited using the context menu.
12. **Destination:** Toggles the Viz scene-specific action's destination property.
13. **Container Path/Actor ID:** Toggles Viz scene-specific action's scene path and Unreal specific actor IDs.
14. **1-Click Action:** In case the **Action Toolbar** is switched off in On Air mode, this options allows you to toggle the 1-Click action behavior.
15. **1-Click Group:** In case the **Action Toolbar** is switched off in On Air mode, this options allows you to toggle the 1-Click group behavior.

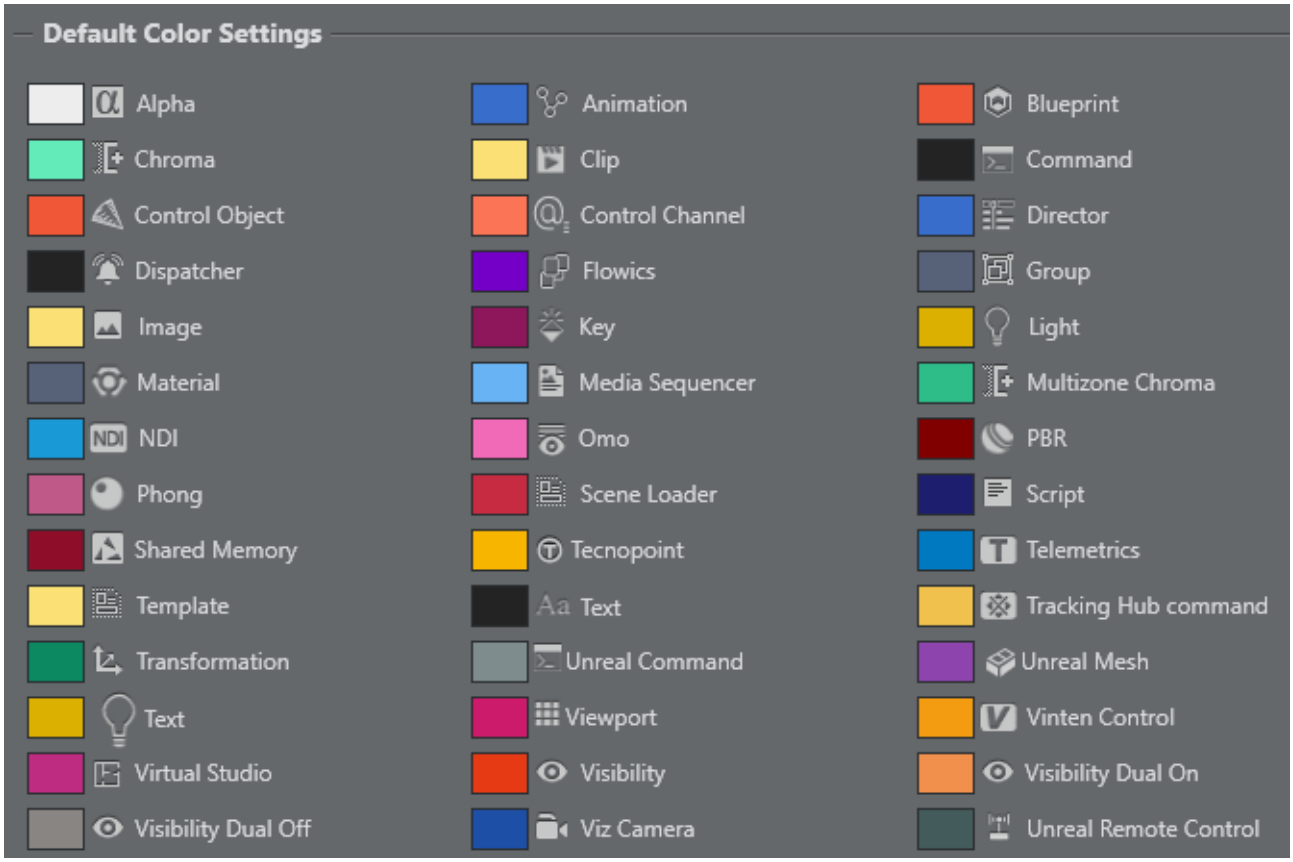
Copyright © 2025 Vizrt



4.2.12 Default Color Setting

You can assign a default color for each action, such as transformation, alpha, visibility and so on. You can still assign a different color in the main area.

Copyright © 2025 Vizrt



See Also

- Project Action Types for more information on how to execute Viz Trio pages.
- GPI Setup for more information about setting up GPI Triggers and Callbacks.
- Video Settings
- Configuring Viz Engine
- Viz Trio User Guide
- http://www.vinten.com/
- Playlist

Copyright © 2025 Vizrt

4.2.13 Configuring Viz Engine

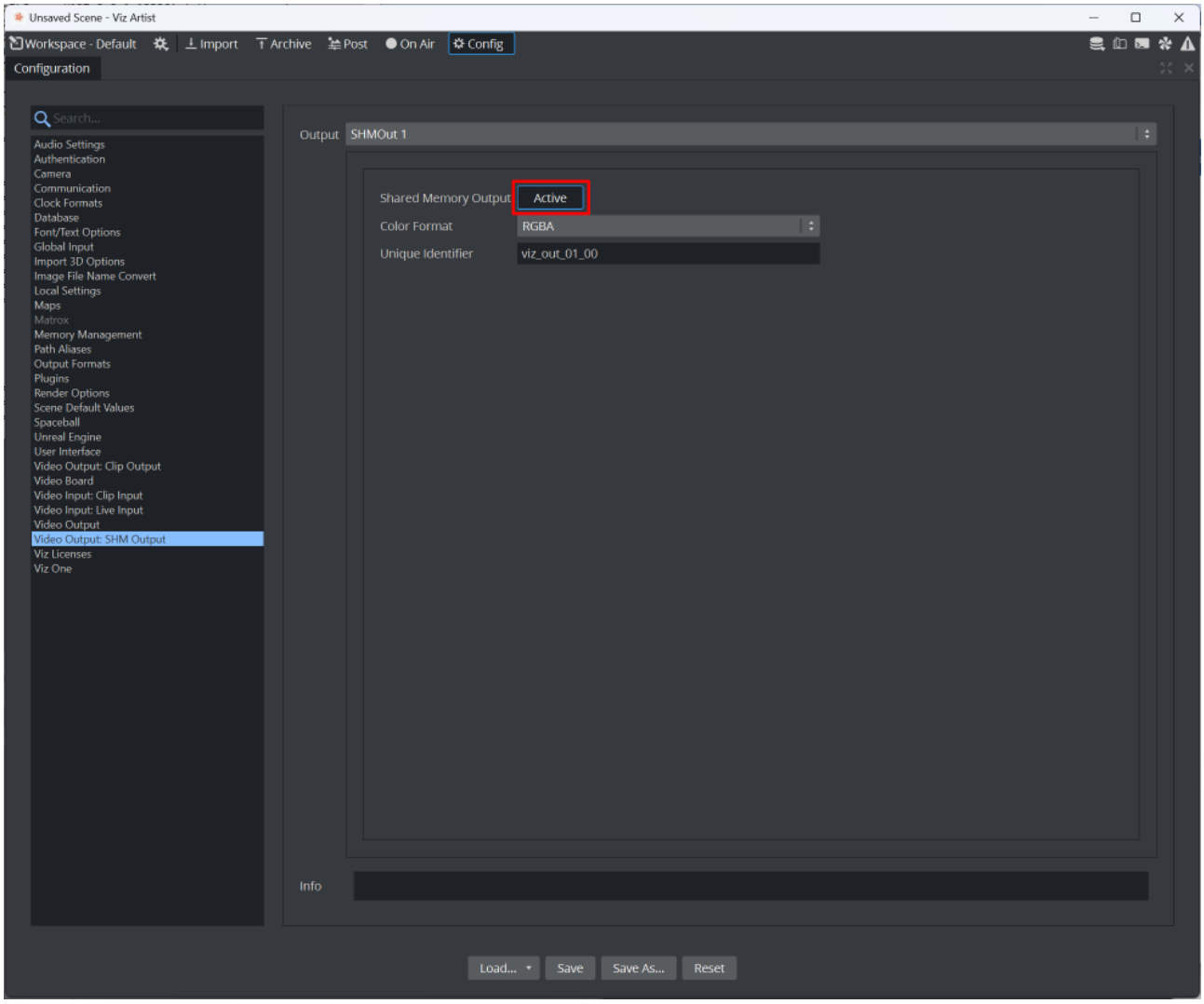
In order to see a preview in Viz Arc, the following three settings in Viz Config must be configured:

- Video Output
- Video Board

> **Note:** Please verify that your system meets the version requirements.

In Viz Engine, go to the **Viz Configuration** tab and select:

1. Video Output SHMOutput: Set **Shared Memory Output** option to **Active.**

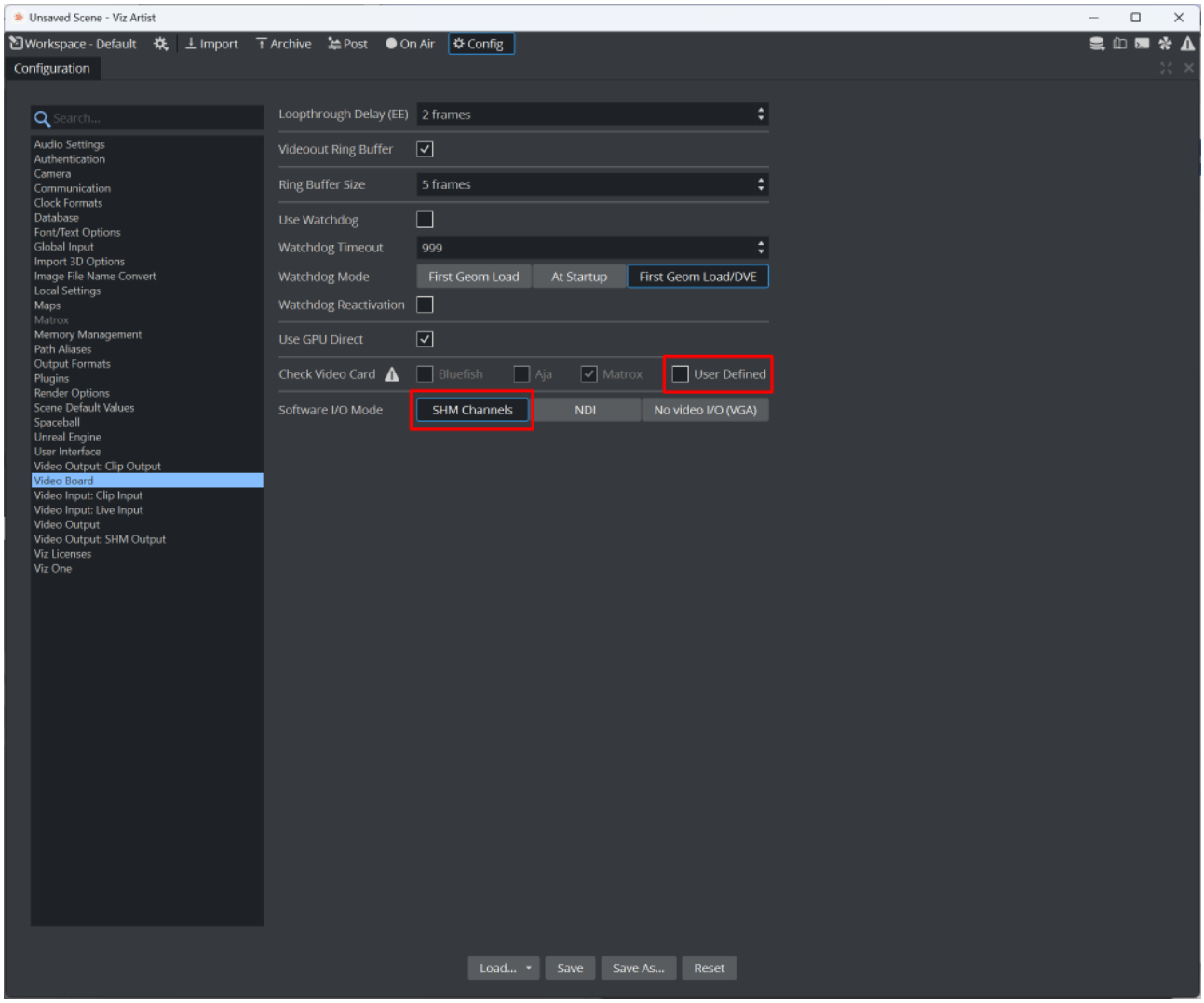


2. Video Board: For non-Matrox configurations, you must configure the following:

- **Software I/O Mode:** Make sure **SHM Channels** is selected.
- **Check Video Card:** Must be set to: **User Defined.**

For Matrox systems, just check the **Matrox** check box.

Copyright © 2025 Vizrt



See Also

- Viz Engine Administrator Guide for more details on the different configuration options for your version of Viz Engine.

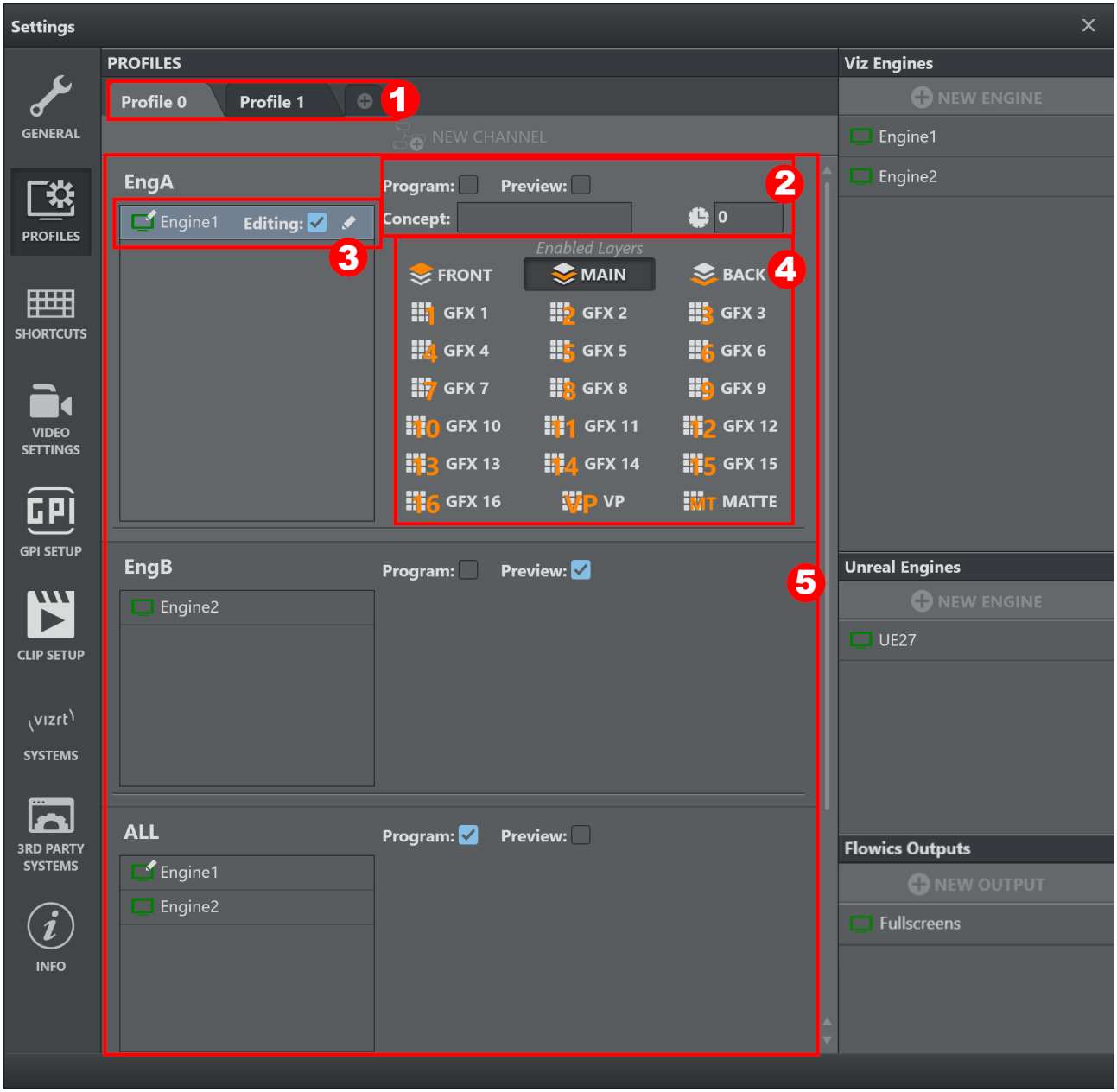
Copyright © 2025 Vizrt

4.3 Profiles

1. **Profiles:** Used to configure the different groups of *Viz* or *Unreal Engines* available for control by the user; these groups are called **Channels.**
2. **Engine and Channel Settings:** Used to configure Program/Preview, the concept and an execution delay each Engine.
3. **Engines list:** Viz Arc is capable of controlling more than one Viz Engine at the same time. For example, when working with multiple tracked cameras. This menu contains all the Engines assigned to a given channel.
4. **Enabled Layers:** Layers enabled in the Scene Tools when selecting a channel and selecting into which layer a scene is loaded.
5. **Channels:** A channel is a group of Viz Engines (or *Unreal Engines*) that receive the same commands at the same time. For example, when executing actions or loading a scene.

To access the Profile config menu, press the **Profiles** icon on the left of the window in the **Config** panel:

Copyright © 2025 Vizrt



This section covers the following topics:

- Profile Configuration
  - To Add a New Profile
  - To Rename a Profile
  - To Delete a Profile
- Engine Configuration
  - Add a Viz Engine
  - To Add an Unreal Engine
  - To Add a Flowics Output
  - To Rename or Edit an Engine
  - Configure and Engine settings

Copyright © 2025 Vizrt

    - To Delete an Engine
- Channel Configuration
    - To Add a Channel
    - To Assign Engines to Channels
    - To Rename a Channel
    - To Delete a Channel
    - To Enable Available Viz Layers
    - To Select Unreal Engine Version
    - To Select a Profile

Note: To connect Viz Arc to an *Unreal Engine* you need to copy the *VizArc* folder to *\Epic Games\UE_4.20\Engine\Plugins*.

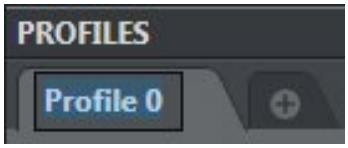
4.3.1 Profile Configuration

To Add a New Profile



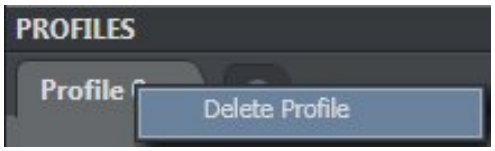
Press the **CREATE PROFILE** icon in the middle of the window or the plus icon  next to the existing profiles.

To Rename a Profile



1. Double-click on the profile name.
2. Insert the desired name.
3. Click **ENTER.**

To Delete a Profile



1. Right-click on the profile name.
2. Select **Delete Profile** in the context menu.

Copyright © 2025 Vizrt

4.3.2 Engine Configuration

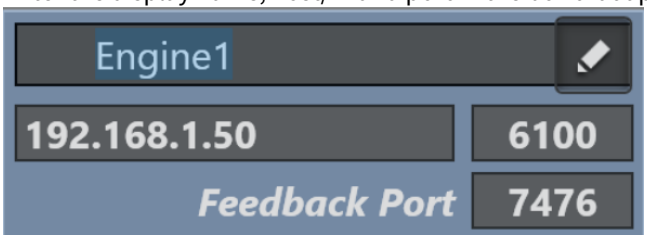
> **Note:** The Engines List is global for *all* Profiles configured in Viz Arc.

Add a Viz Engine



1. Press the **NEW ENGINE** icon on the right of the window or the **add** icon 

at the top of the Viz Engines list.
2. Enter the display name, host/IP and port in the box that appears.



   The Feedback Port is the `command_feedback_port` as configured on the VizEngine (default `7476`).
3. Press **ENTER**.

To Add an Unreal Engine



1. Press the **NEW ENGINE** icon on the right of the window or the **add** icon 

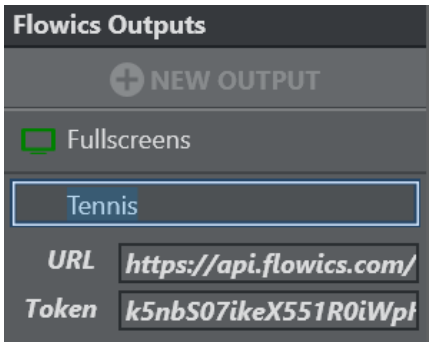
at the top of the Unreal Engines list.
2. Enter the display name, host/IP and Viz Arc Unreal Engine plug-in port and the Launcher Port in the boxes that appear.



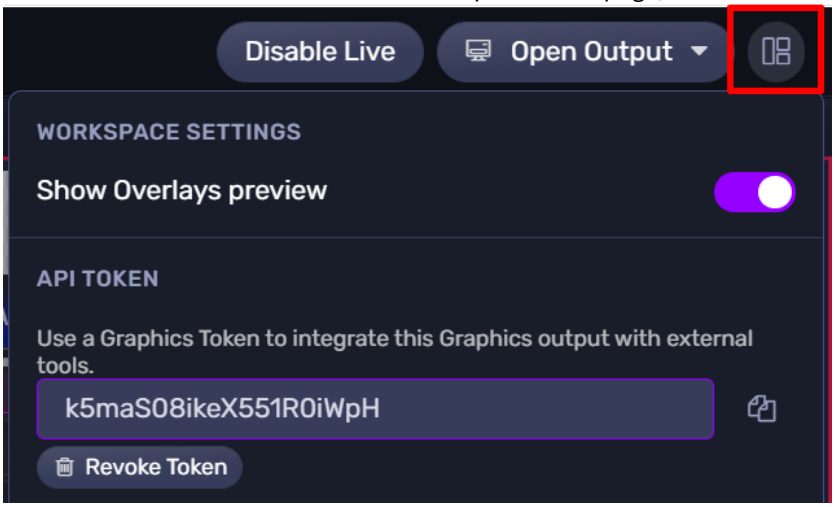
Copyright © 2025 Vizrt

3. Press **ENTER.**

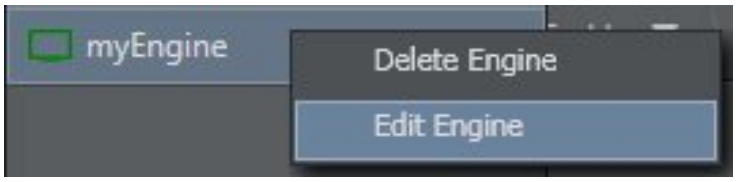
To Add a Flowics Output



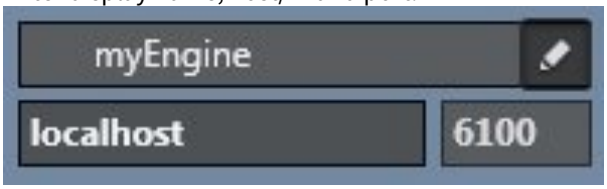
1. Press the **NEW OUTPUT** icon.
2. Enter the display name of the new output and enter its API token.
   The API token can be found on Flowics output control page, under the settings panel:



To Rename or Edit an Engine



1. Right-click on the Engine name.
2. Select **Edit Engine** in the context menu or press the **Edit** icon next to the Engine name.
3. Enter display name, host/IP and port.



Copyright © 2025 Vizrt

4. Press **ENTER**.

Configure and Engine settings



It's possible to assign every Engine as a **Program** or **Preview** engine.



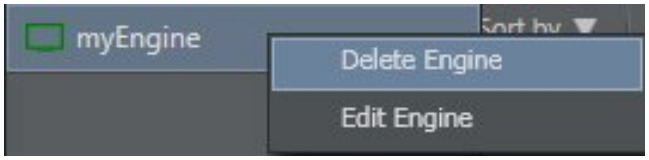
An Engine can also be set as **Editing Engine**, which means the Engine is used for a scene tree load, or to select and objectify a Transition Logic scene, for example.



The **Concept** determines which scene concept is used by the engine when a scene is loaded through the **Scene Loader Action** or a **Template Action**. **The Execution Delay** can be used to delay all commands that are sent to the engine. This can be useful for set-ups involving e.g. AR and Videowall engines where output runtimes of the various subsystems are not uniform.

Warning: Setting an Engine that is On Air into editing mode may load a scene tree while another scene is On Air.

To Delete an Engine



1. Right-click on the Engine name.
2. Select **Delete Engine** in the context menu.
3. Confirm by pressing **Yes**.

4.3.3 Channel Configuration

To Add a Channel



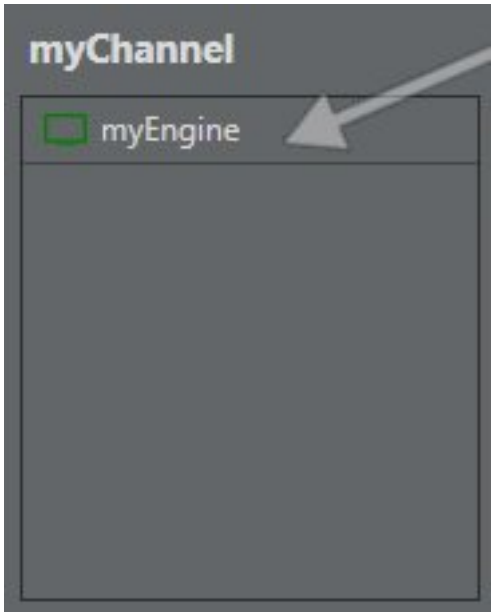
Press the **NEW CHANNEL** icon in the middle of the profiles section or the **Add** icon 

at the top of the Channel's list.

Note: It's possible to create multiple channels and assign a unique name to each channel.

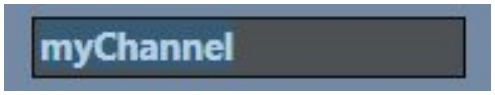
Copyright © 2025 Vizrt

To Assign Engines to Channels



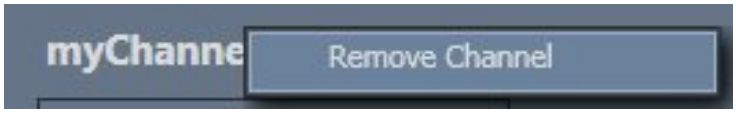
Drag and drop Engines from the Engines list *(right).*

To Rename a Channel



1. Double-click on the channel name.
2. Enter the desired name.
3. Press **ENTER.**

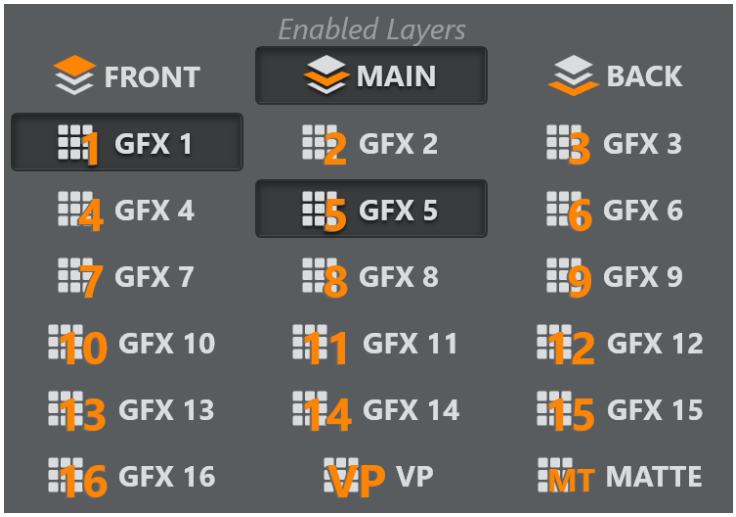
To Delete a Channel



1. Right-click on the channel name.
2. Select **Remove Channel** in the context menu.
3. Confirm by pressing **Yes**.

Copyright © 2025 Vizrt

To Enable Available Viz Layers

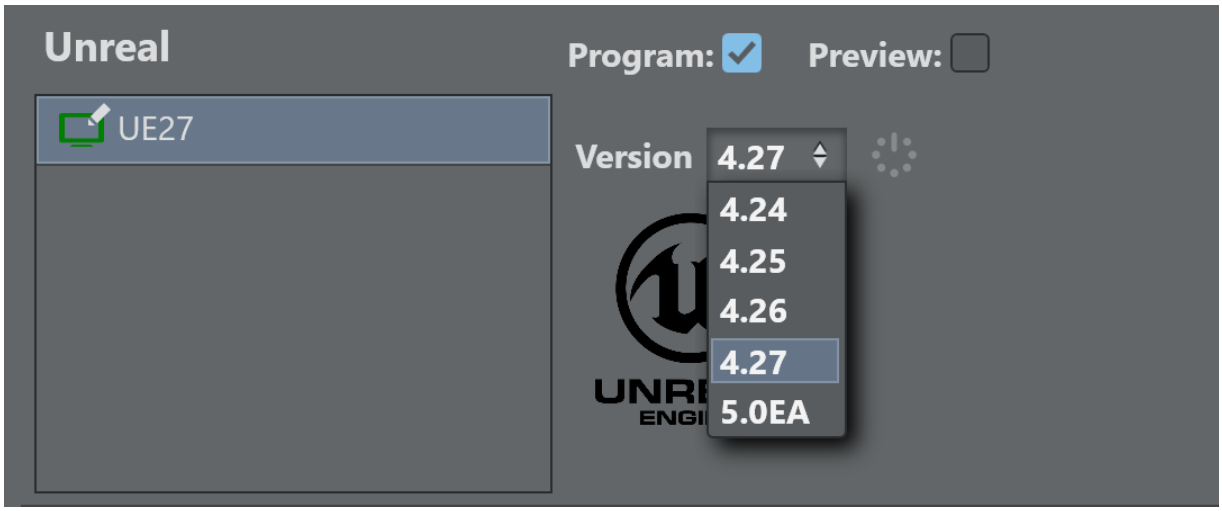


Select one or more layers by pressing them in the **Channel** menu.

> **Note:** Layers let you send scenes and commands to different output levels within Viz Engine. The main layers are FRONT LAYER , MAIN LAYER and BACK LAYER. graphics are usually sent to the MAIN LAYER. If you want to view a graphic On Air you need to have at least one layer enabled.

To Select Unreal Engine Version

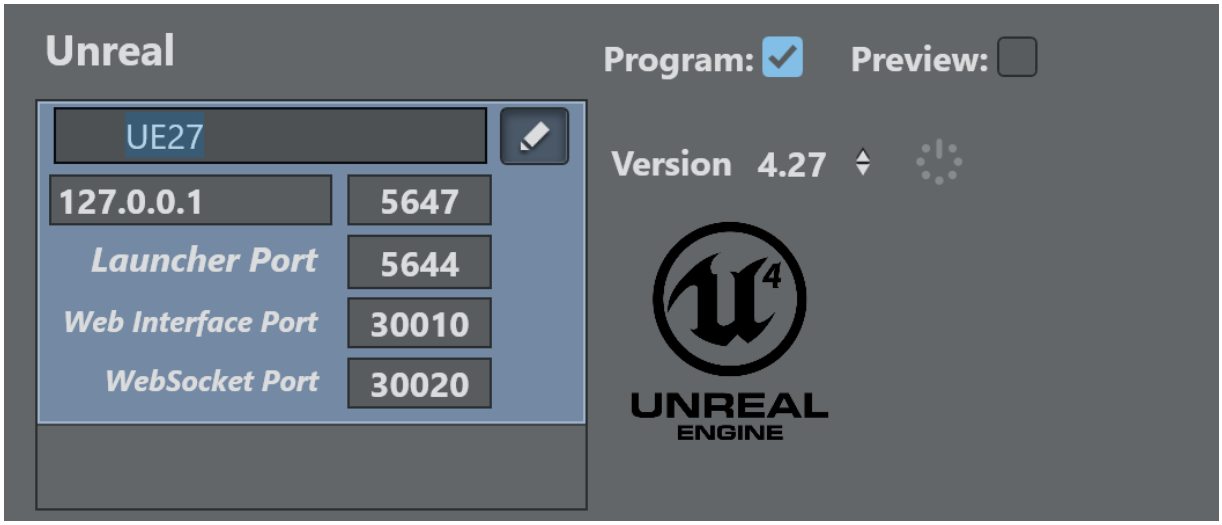
If the channel consists of one or more Unreal Engines, the **Channel** menu looks like this:



For each Engine, select the Unreal Version to be used by selecting it from the drop down menu. The adjacent **start/restart** button starts the respective Unreal version.

> **Note:** The **Launcher Port** must be configured properly and the Viz Arc Unreal Loader service must run on the target machine. See Unreal Loader for more on setting up and configuring Loader service.

Copyright © 2025 Vizrt

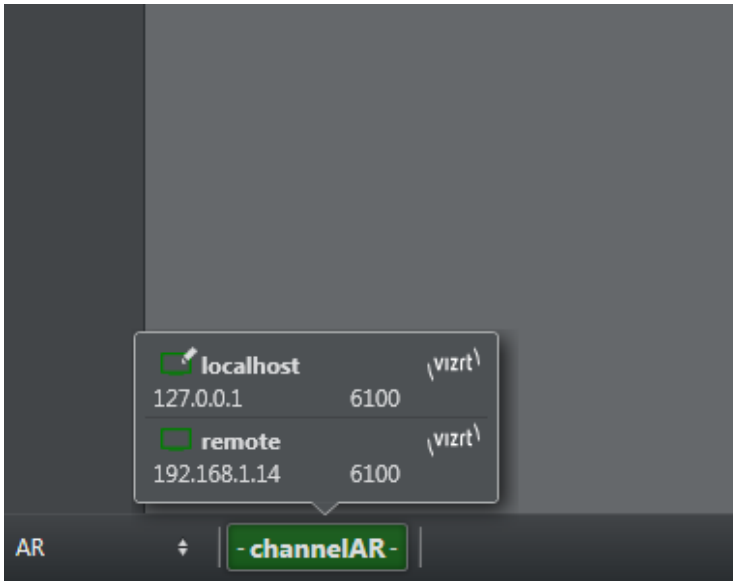


- **Host/Port**: The hostname or IP address running the Unreal Engine and the port on which the Viz Arc Unreal plug-in is configured to listen to.
- **Launcher Port**: The port used by the Viz Unreal Loader service.
- **Web Interface Port:** The port where UE exposes the Remote Control API (UE Version >= 4.26 only).
- **WebSocket Port:** The web socket port where UE exposes the Remote Control Web Interface (UE Version >= 4.26 only).

WebSocket Port and **Web Interface Port** are only relevant when the UE Preset actions are used.

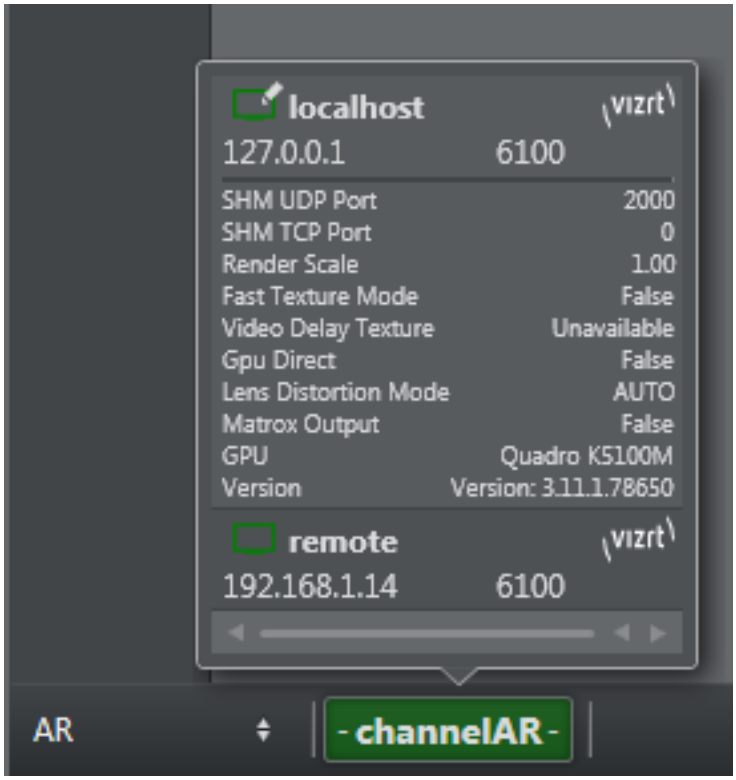
To Select a Profile

Once you have created a profile, you can select it from the drop down menu at the bottom left corner of the UI. All channels containing the selected profile are displayed on the right.

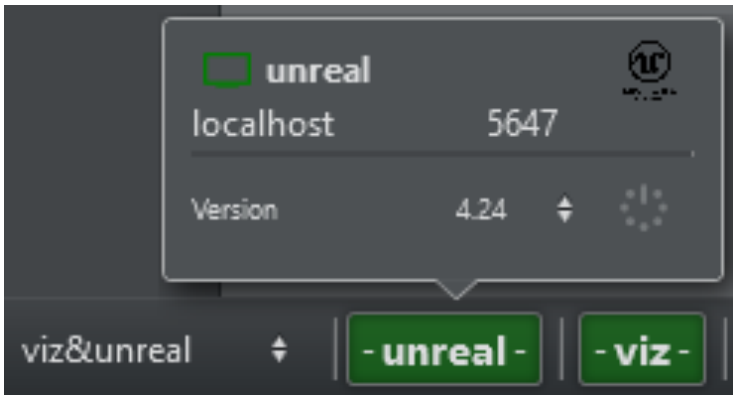


Click on a channel to see which Engines it contains.

Copyright © 2025 Vizrt

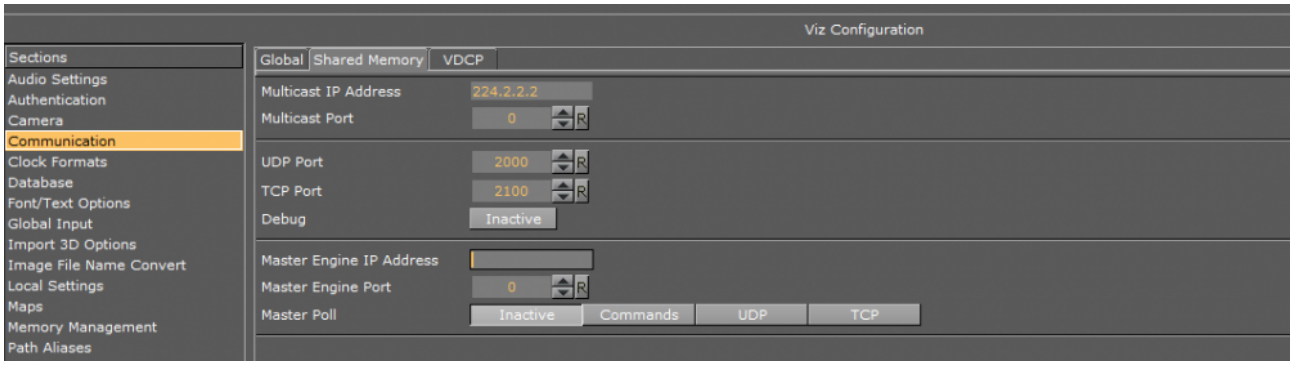


If the channel is an Unreal channel, you can select a different version to run or start/restart the Unreal Engine.



Click on the small triangle at the right of each Engine to see more details about the current Engine configuration. The first two entries show the shared memory communication port settings, which can be used with scripting or SHM. To configure the Shared Memory ports on Viz Engine, go to Viz Engine configuration under **Communication > Shared Memory** where you can configure UDP and TCP ports for Shared Memory updates.

Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

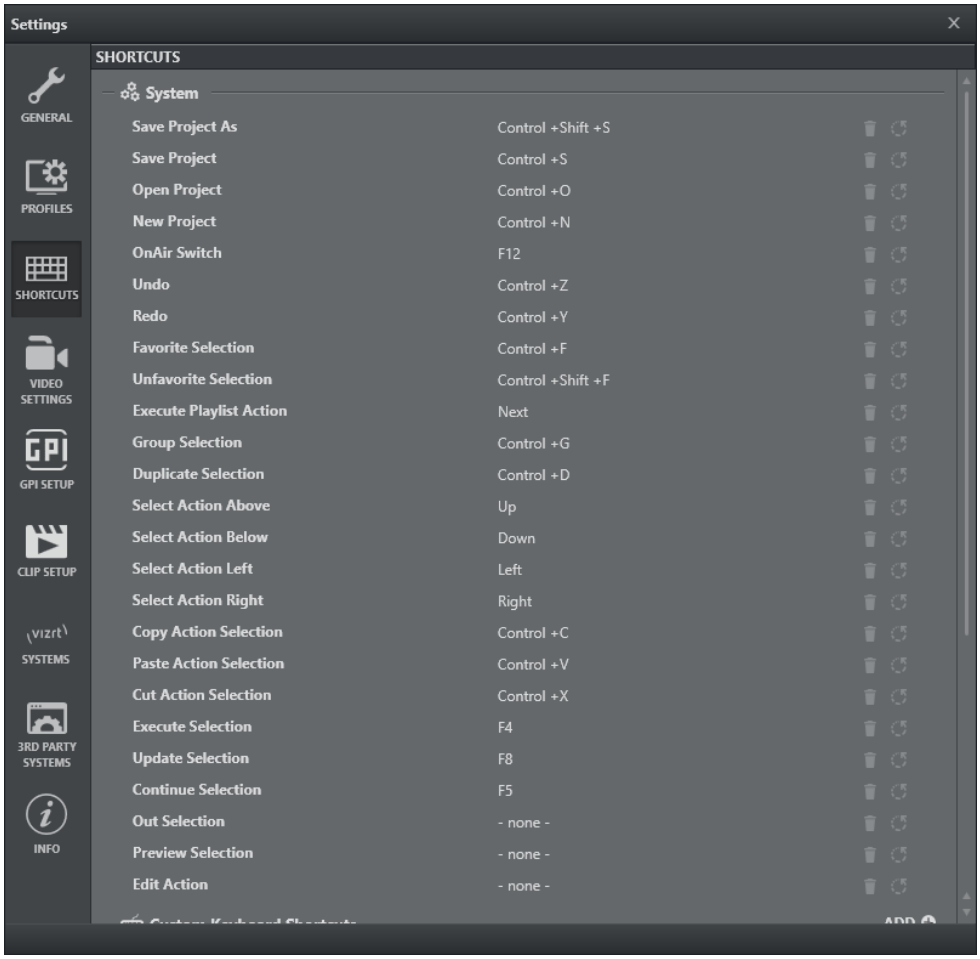
4.4 Keyboard Shortcuts

To access the **Keyboard Shortcuts** configuration menu, select the Keyboard Shortcuts icon on the left of the window in the **Config** panel.

4.4.1 Keyboard Shortcuts

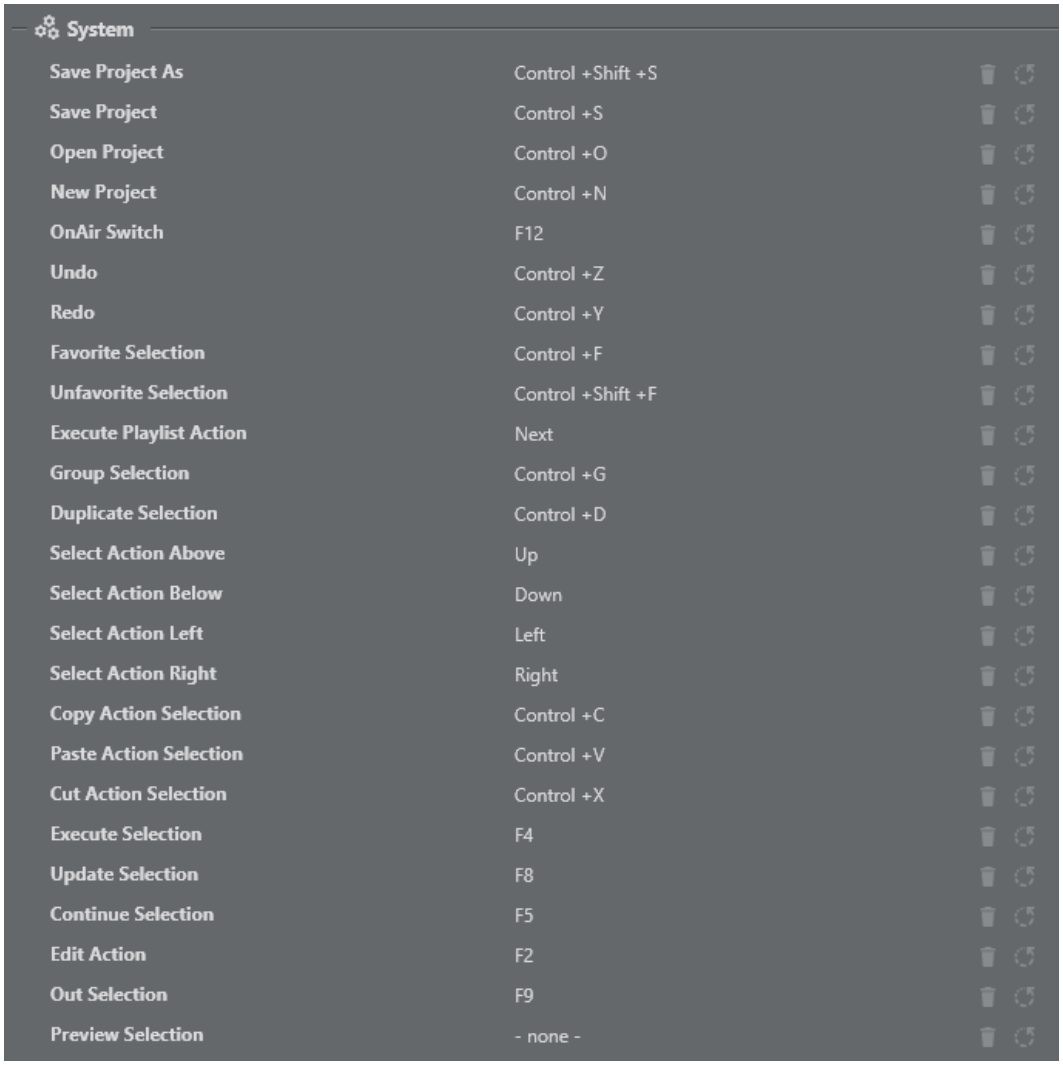
Keyboard Shortcuts are divided into three sections:

- System Shortcuts
- Custom Keyboard Action Shortcuts
- Custom Stream Deck Shortcuts



Copyright © 2025 Vizrt

System Shortcuts

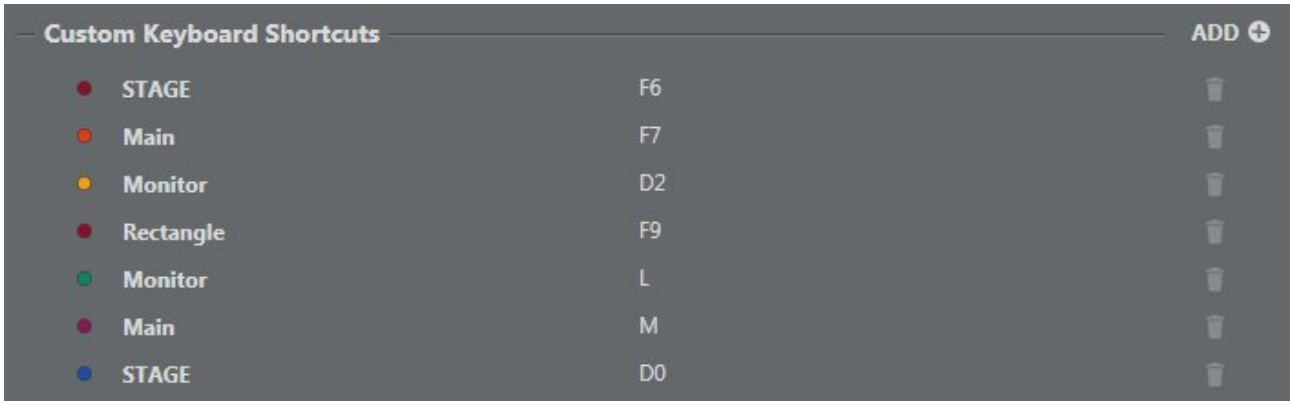


System Shortcuts apply operations that are are global to the Viz Arc system, such as **Open Project**, **Save**, **Undo**, **Redo**, etc. System operations that can be controlled with keyboard shortcuts are listed at the top of the window. It's possible to change the default shortcut for an existing system action, remove the shortcut, or reset it back to default.

All modified system shortcuts data is stored in the *keybinds.json* file. For more details, see Files and Folders.

Copyright © 2025 Vizrt

Custom Keyboard Action Shortcuts



Action Shortcuts are used to assign a key or combination of keys to execute actions. This menu contains a list of all actions that have a keyboard shortcut assigned and the action box color for the action.

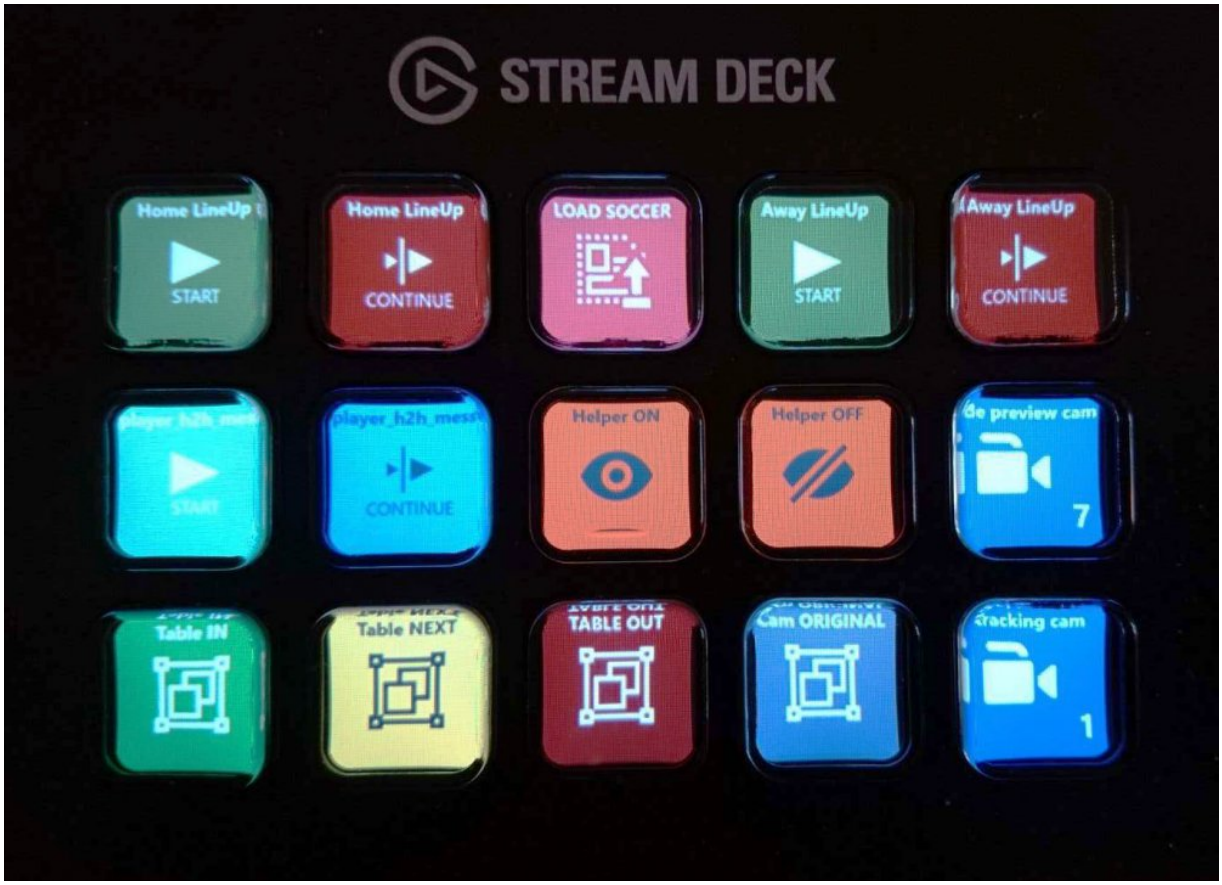
It's possible to assign keyboard shortcuts to any Action in a project and to assign the same keyboard shortcut to more than one action.
You can add Custom Action Shortcuts directly from this window.

Custom Stream Deck Shortcuts



Custom Stream Deck Shortcuts are used to assign an action (or group) to a specific Stream Deck button (if connected). This menu contains a list of all actions that have a keyboard shortcut assigned and the action box color for the action.

Copyright © 2025 Vizrt



Note: All action shortcuts are stored as part of the *project.json* file. For more details, see Files and Folders.

4.4.2 To Manage Shortcuts

Control +Shift +S

1. Click on the **Shortcut** item.
2. Enter the desired key combination on your keyboard.

To Remove Shortcuts

Press the **Trash can** icon next to the related shortcut.

To Reset Shortcuts to Default

Press the **Reset** icon next to the desired shortcut (*for system keyboard shortcuts only*).

4.4.3 To Add Custom Action Shortcuts

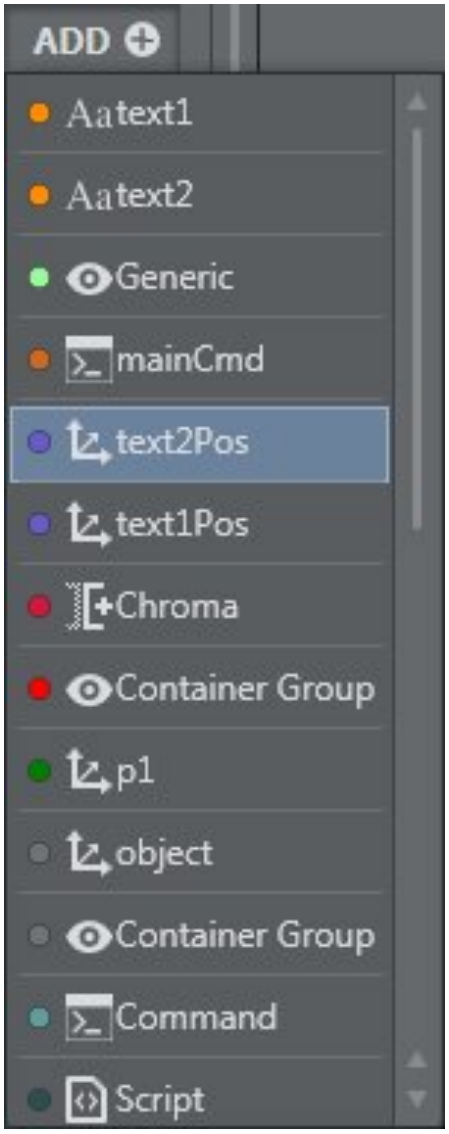
There are two ways to add an action shortcut:

- Config Window

Copyright © 2025 Vizrt

- Action Window

Config Window



1. Click the **Add** button in the **Config > Keyboard Shortcuts** menu.
2. Select an action from the available options. A new Action row appears in the window.



3. Click on the shortcut area and press the desired keyboard shortcut. It's possible to add both Keyboard and Stream Deck (if available) custom shortcuts.

Copyright © 2025 Vizrt

Action Window



In the action **Edit** menu, click on the shortcut area and press the desired keyboard shortcut. It will automatically be added to the **Custom Keyboard Shortcuts** list.

See Also

- Actions View to learn more about assigning keyboard shortcuts to actions.
- https://www.elgato.com/en/gaming/stream-deck for more information about Stream Deck hardware.

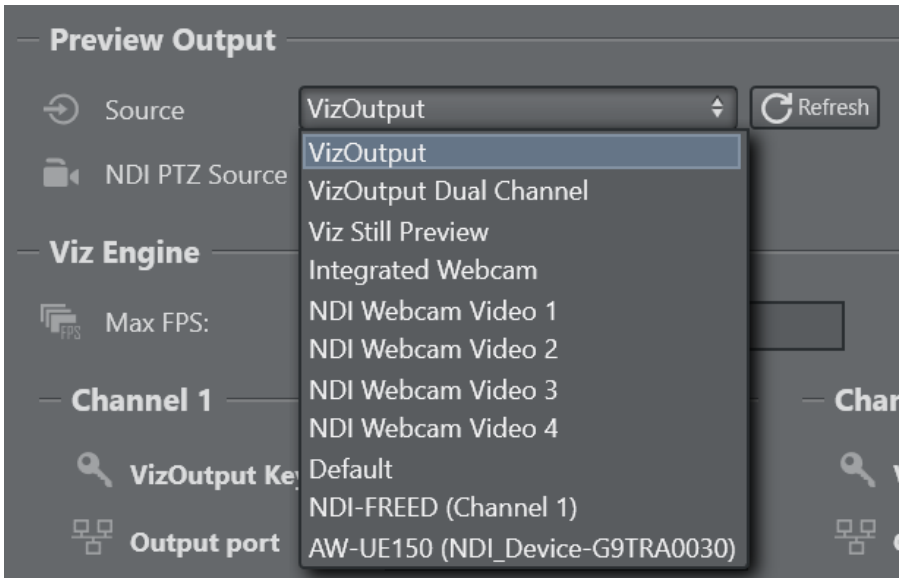
Copyright © 2025 Vizrt

4.5 Video Settings

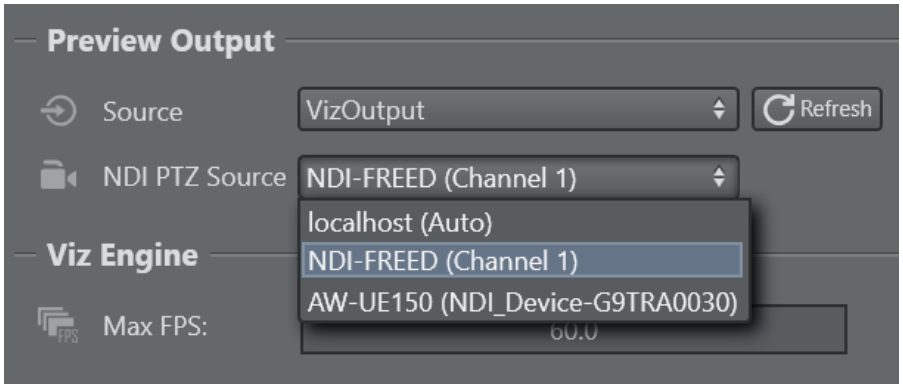
4.5.1 Preview Output

Preview Output lets you choose between the following outputs:

- Local Viz Output
- Still Preview
- Direct Show Sources
- NDI



The selected source is displayed throughout all Viz Arc interfaces. Use the **Refresh** button to force the detection of newly connected sources.



The **NDI PTZ Source** can be selected explicitly to any NDI source with PTZ capabilities or use **localhost (Auto)** to use the same source as selected in **Source**.

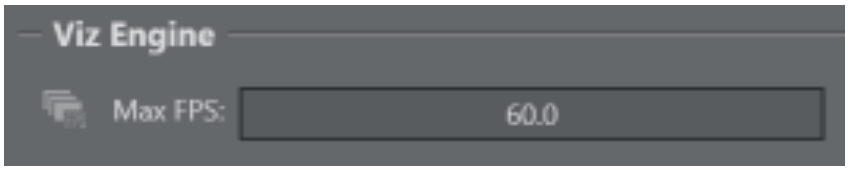
Use the **Refresh** button on the source to update those sources. The selection enables a PTZ control overlay on the preview output.

Copyright © 2025 Vizrt



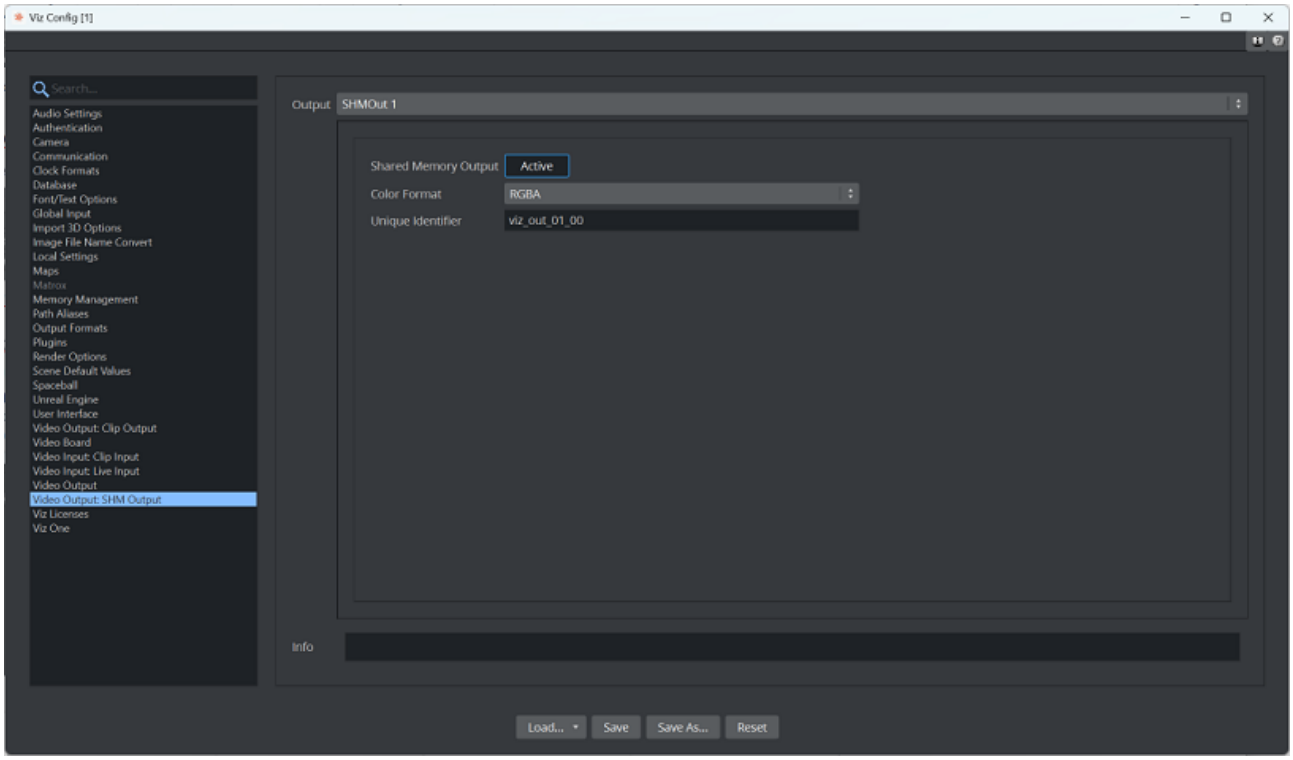
VizOutput

The Viz Arc built-in preview uses the Viz Engine's Shared Memory communication to display the output of Viz Engine.



- **Max FPS:** Limit the maximum number frames per second displayed for the Viz Output, sometimes useful when experiencing performance issues.

Copyright © 2025 Vizrt



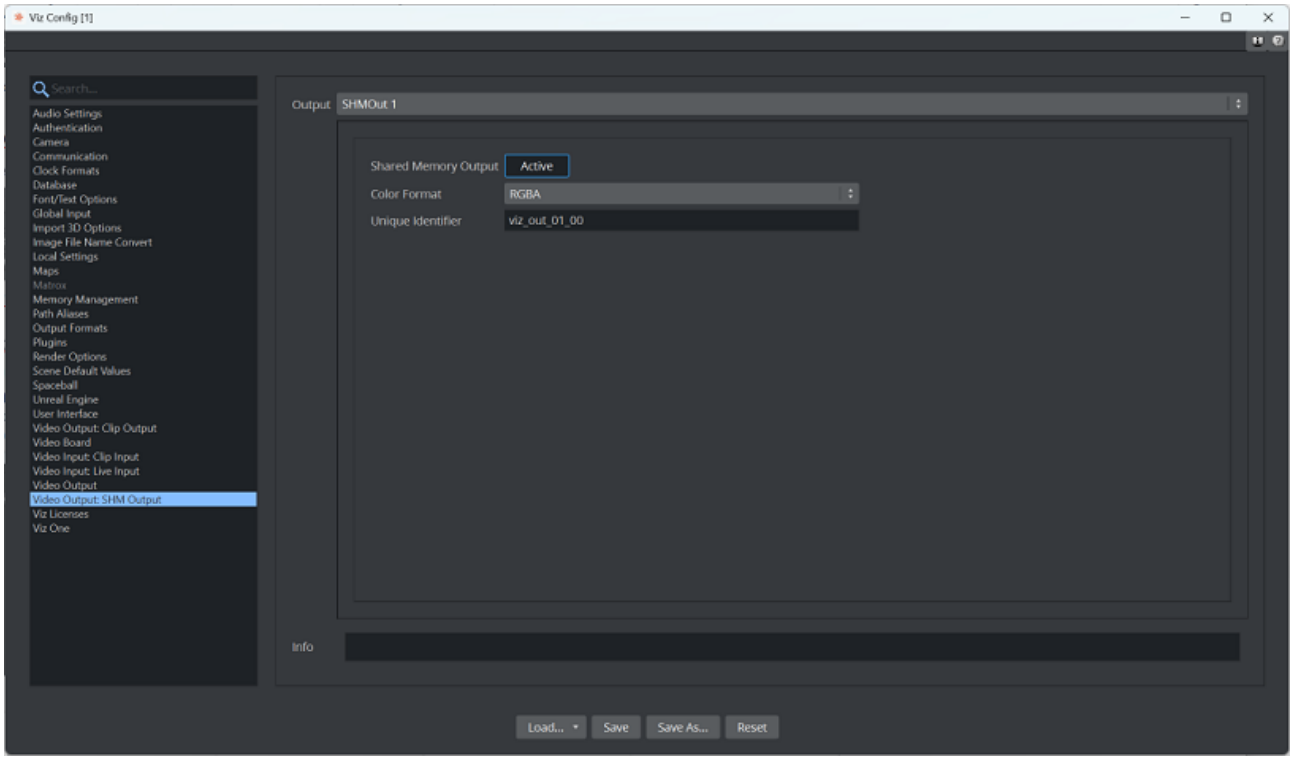
It's also possible to configure VizOutput settings for Channel 1 and Cannel 2:

- **VizOutput Key:** The VizOutputKey Sync Key is used for receiving the *Viz Preview.* You must configure the desired key in the Viz Engine configuration (*default key: viz_out_01_00*).
- **Output port:** The default Viz Engine TCP communication port.
- **VizOutput Sync Key:** The value must match the **SHMSyncMasterKey** Viz Engine config setting (*default key: viz_engine_SHMsync*).

To have a working preview, you must

1. Install Viz Engine as a video machine even if you do not have video hardware installed on your system.
2. Configure VizOutput settings in Viz Arc, in the **Video Settings** section.
3. Configure your Viz Engine to also output through Shared Memory.

Copyright © 2025 Vizrt



Important! You must also check your Viz Engine Configuration.

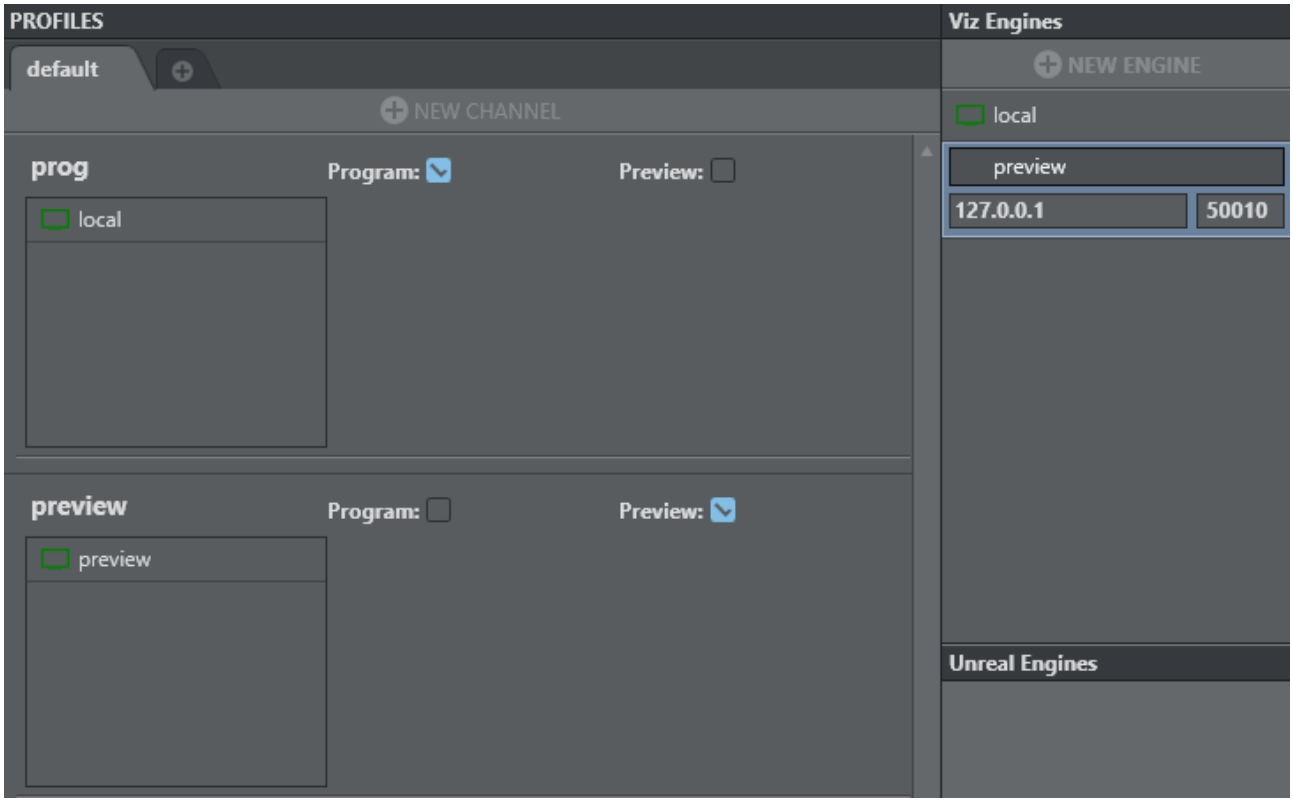
For more information, see Preview Output.

Viz Still Preview

Note: This setting is intended for Viz still preview only.

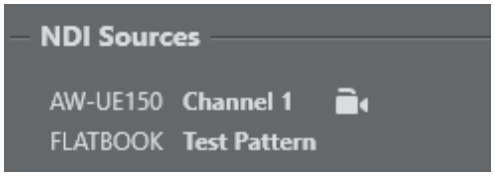
To configure still preview, configure a dedicated Channel in the Profiles section and tag it as **Preview Channel**. The communication port must be configured to match the Viz Engine's still preview port, which is by default `50010`. A still preview configuration example is shown below:

Copyright © 2025 Vizrt



The still preview is used in conjunction with the template workflow and can be generated using the **Preview Selection** keyboard shortcut.

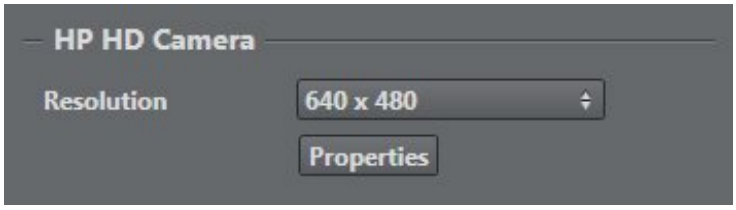
NDI



All detected NDI sources are listed here. The  icon indicates that the source is a controllable PTZ source.

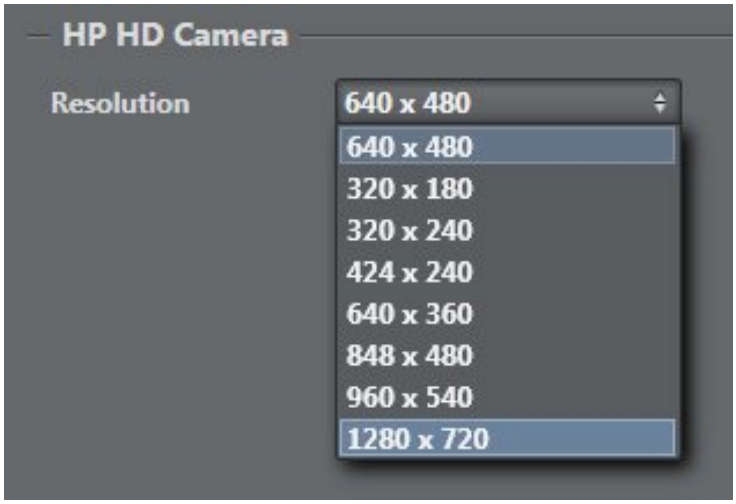
Direct Show

Viz Arc displays all deleted DirectShow video sources. An example of configuring an external device (HP HD Camera) connected to Viz Arc is shown below:

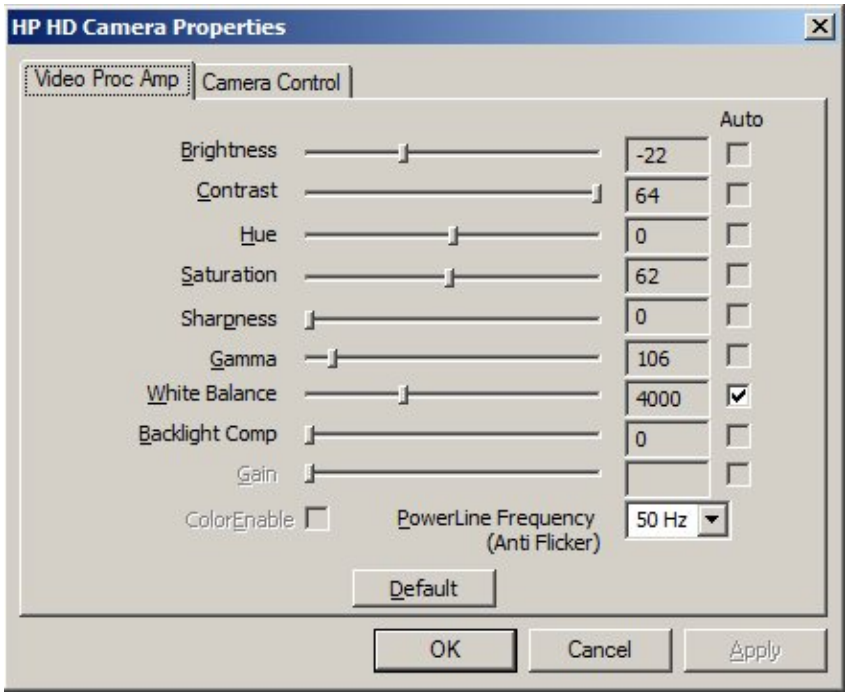


Having selected HP HD Camera, select the resolution required:

Copyright © 2025 Vizrt



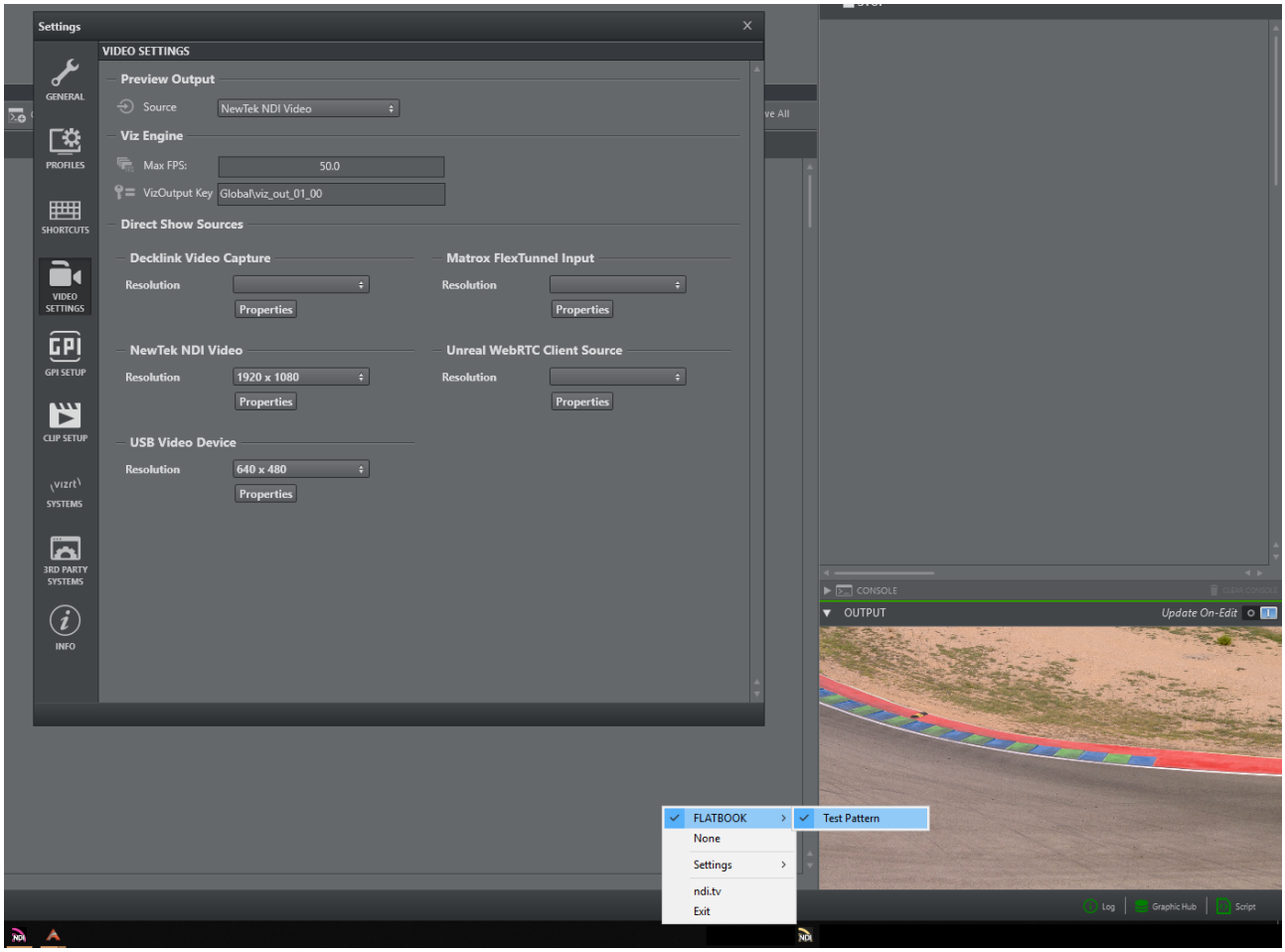
Press the **Properties** button. A popup window appears where you can customize output properties.



NDI through DirectShow

DirectShow sources lets you visualize any NDI stream within the system. The NDI tools must be installed on the system and the NDI Virtual Input needs to run. From there, select the desired NDI stream. In Viz Arc, select the **NewTek NDI Video** filter. No further configuration is required on the source filter itself.

Copyright © 2025 Vizrt

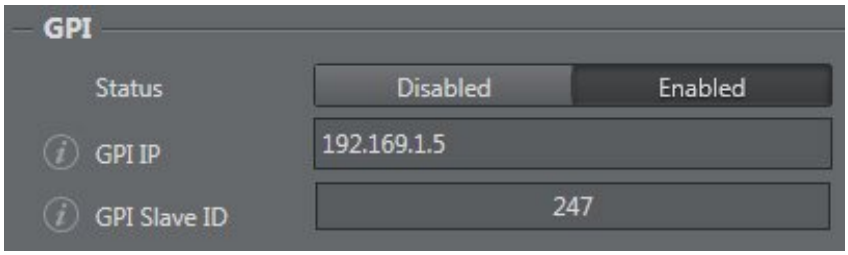


Copyright © 2025 Vizrt

4.6 GPI Setup

4.6.1 GPI

It's possible to trigger actions using GPI in Viz Arc. Configure settings related to GPI using the menu below:

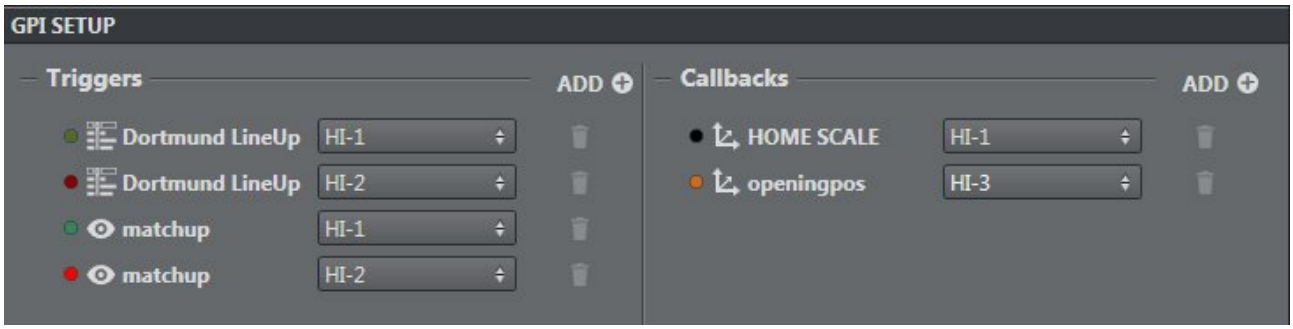


- **GPI IP:** IP port that communicates with GPI.
- **GPI Slave ID:** Defines the Slave ID of the Ethernet box.

Note: Currently, only the Sealevel SeaI/O-410E board is supported.

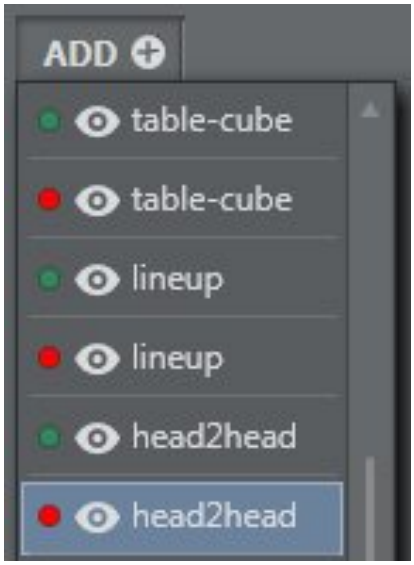
4.6.2 Triggers

All action boxes set to **Trigger** and/or **Callback**, and their related channels (*HI-0/HI-15, LO-0/LO-15*) are shown in the menu below:



It's possible to add every action box set in the main area as **Trigger** and/or **Callback:**

Copyright © 2025 Vizrt



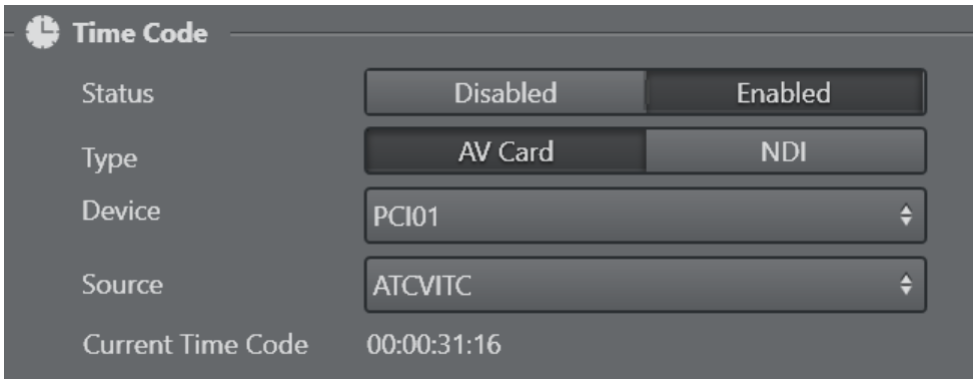
To remove an element, click the **trash icon** next to the element you want to delete.

Copyright © 2025 Vizrt

4.7 Timecode Setup

4.7.1 Time Code

Enable or disable Timecode based triggering. Choose between AV Card or NDI Stream as Timecode source.



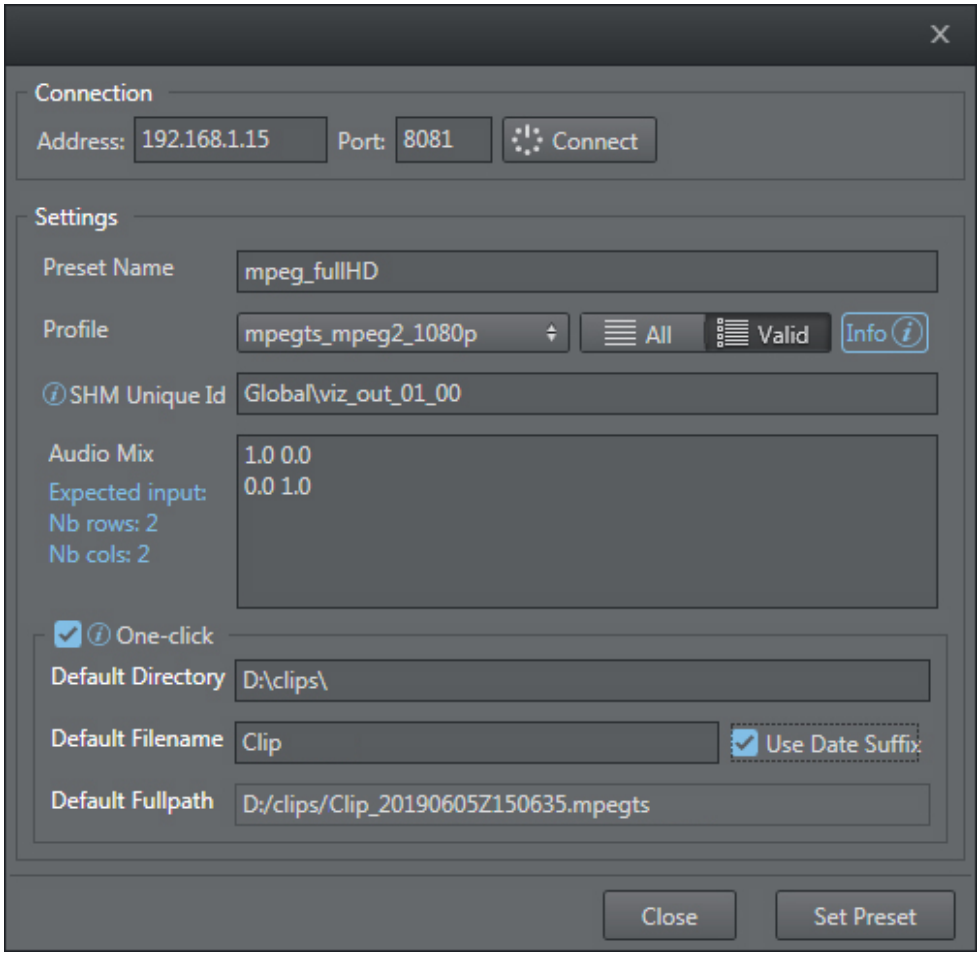
Please refer to the Timecode configuration section for more information.

Copyright © 2025 Vizrt

4.8 Clip Setup

This menu lets you set up presets for Viz Coder or Clip out channel. To add a preset, click the **ADD** ADD icon:

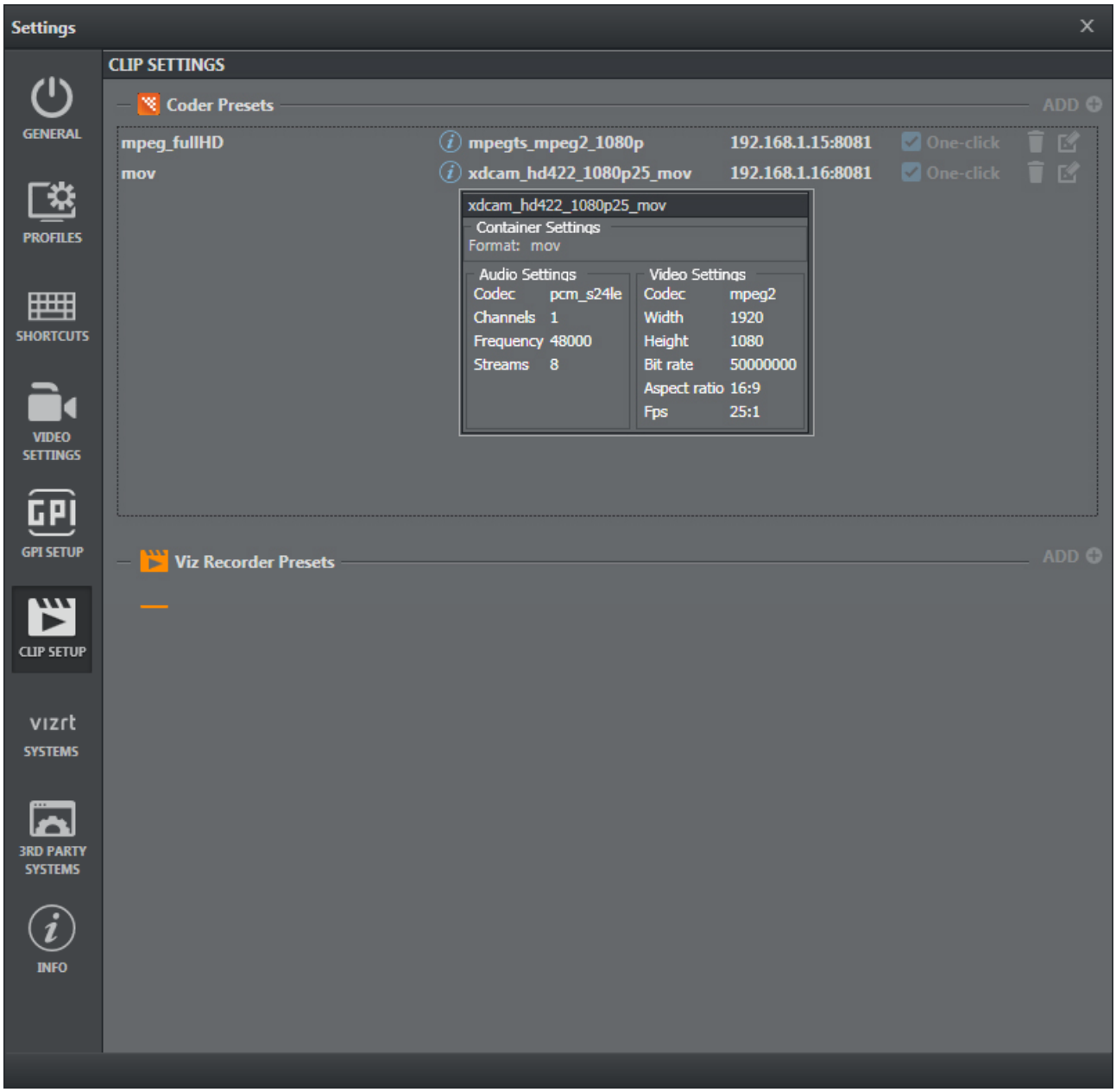
4.8.1 Coder Preset



- To create a valid connection to retrieve preset profiles, insert the correct address and port and press the **Connect** button.
- Once connected, you can use settings to:
  - Insert a custom preset name (if empty, once you select a profile it automatically sets the default name).
  - Select an available profile from a list.
  - Set the SHM unique ID (the same one you configured in Viz Engine). Audio mix contains values that determine the audio settings. The expected preset values are shown on the left.
- A one-click option is available that lets you record files without inserting default settings each time (for example, directory, filename and full path name).

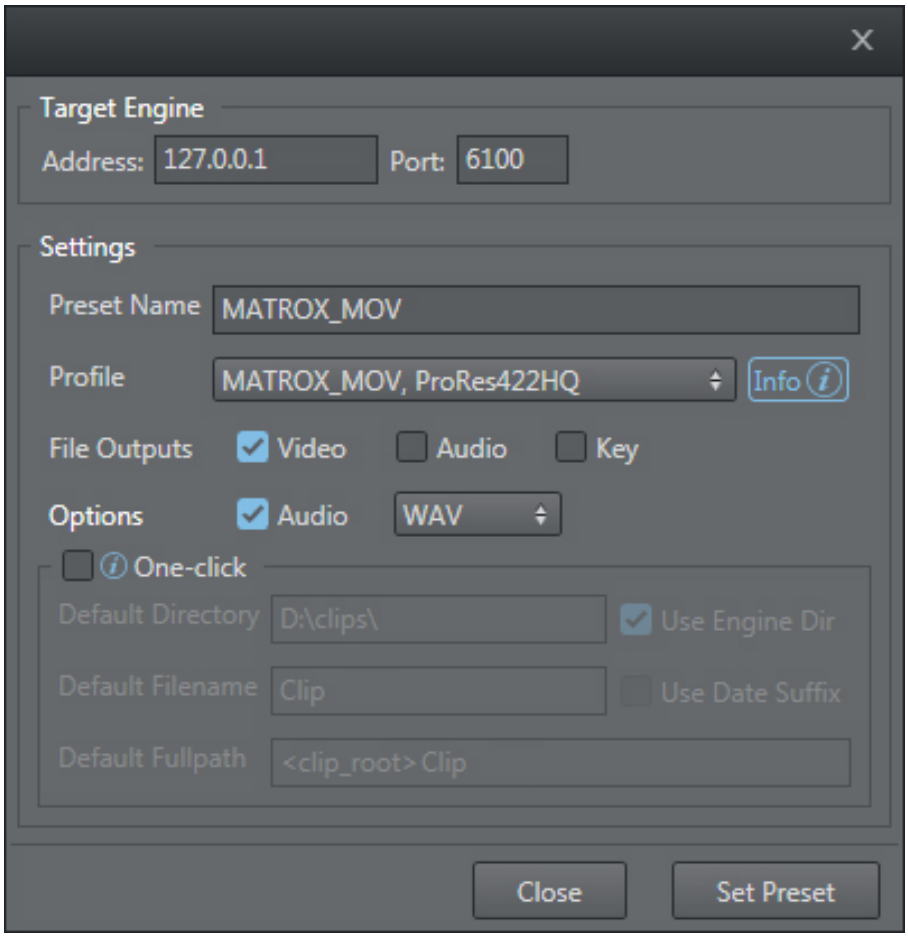
Copyright © 2025 Vizrt

Once created, the new preset is added to the coder preset list. You can remove or modify preset parameters by pressing the respective icon next to the item.



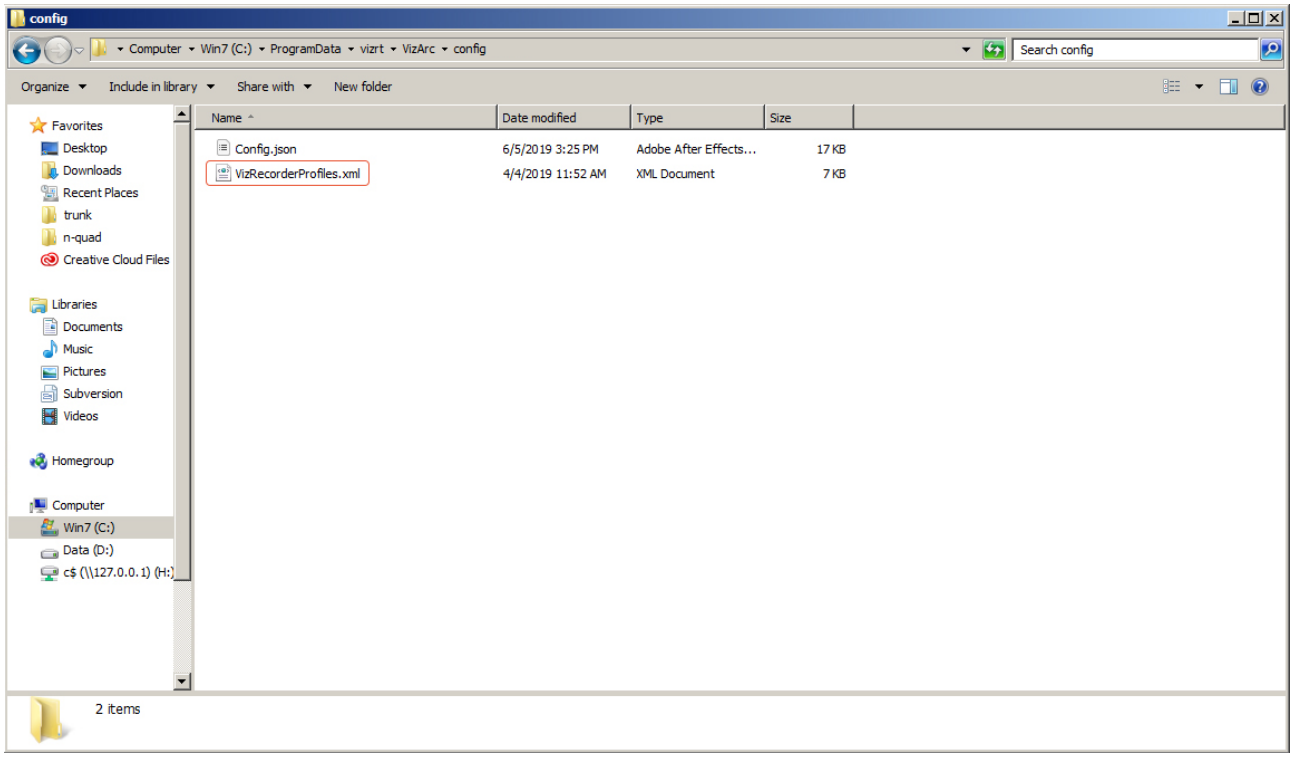
Copyright © 2025 Vizrt

4.8.2 Viz Recorder Preset



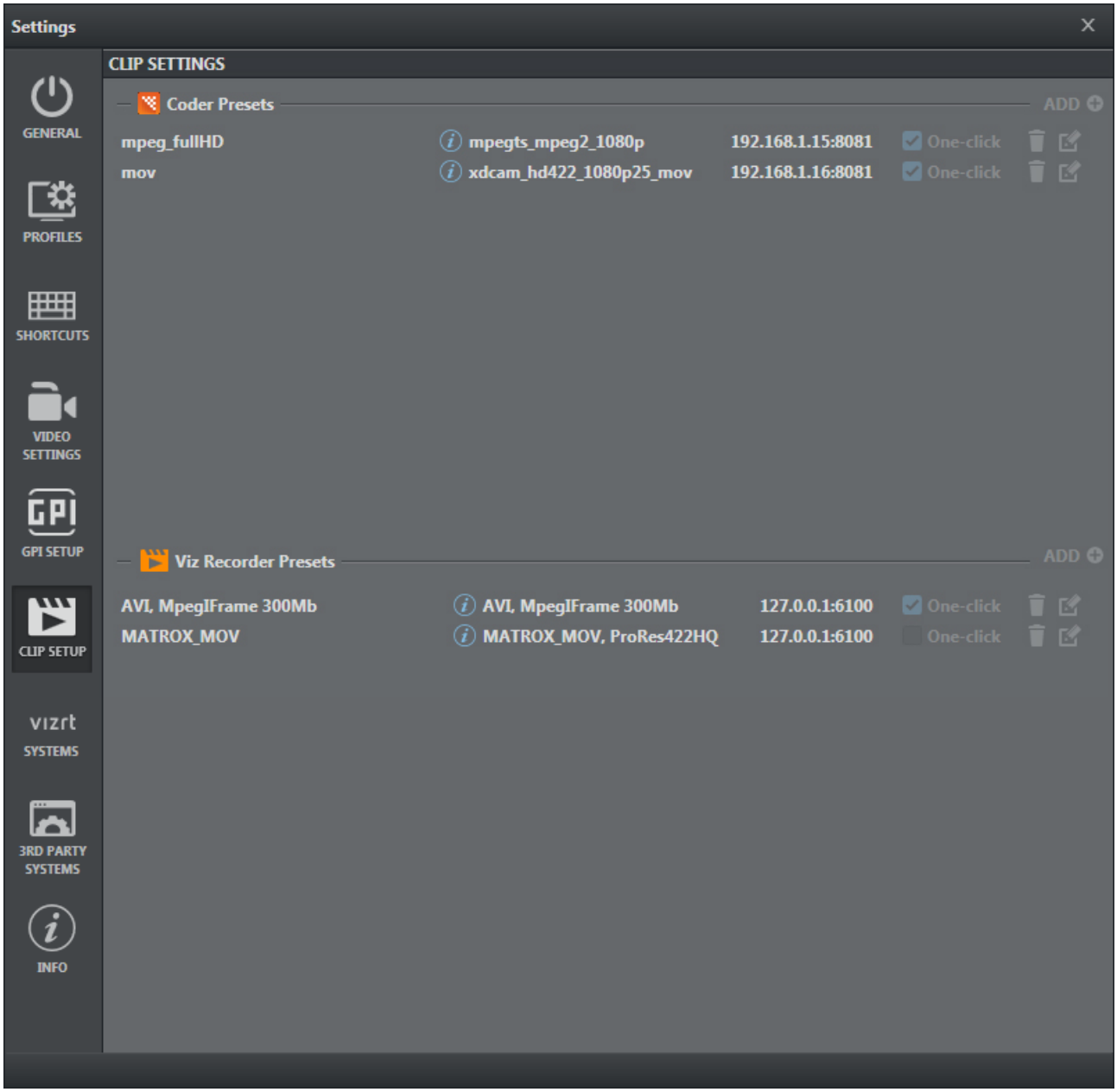
- First, insert the correct address and port to retrieve the profile list.
- Once connected, use the settings menu to:
  - Insert a custom preset name (once you select a profile it automatically sets the default name).
  - Select an available profile from a list.
  - Choose the file type to export (video, audio and/or key). In the case of video export, you can also associate *.waw*, *.w64* or *.aiff* audio types.
  - A one-click option is available that lets you record files without inserting default settings each time (for example, directory, filename and full path name).

Copyright © 2025 Vizrt



The XML file, called *VizRecorderProfiles.xml,* containing all codecs is found in a list in the *config* subfolder. Here, you can add or modify items in the list.

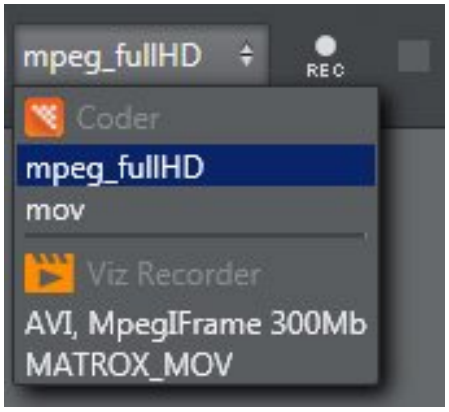
Copyright © 2025 Vizrt



The newly created preset is added to the coder preset list. You can remove or modify preset parameters by pressing the respective icon next to the item.

All presets are displayed in the toolbar combo box.

Copyright © 2025 Vizrt



See Also

- Configuring Viz Engine for more information about configuring Shared Memory.

Copyright © 2025 Vizrt

4.9 Info

The Info window contains general software information:

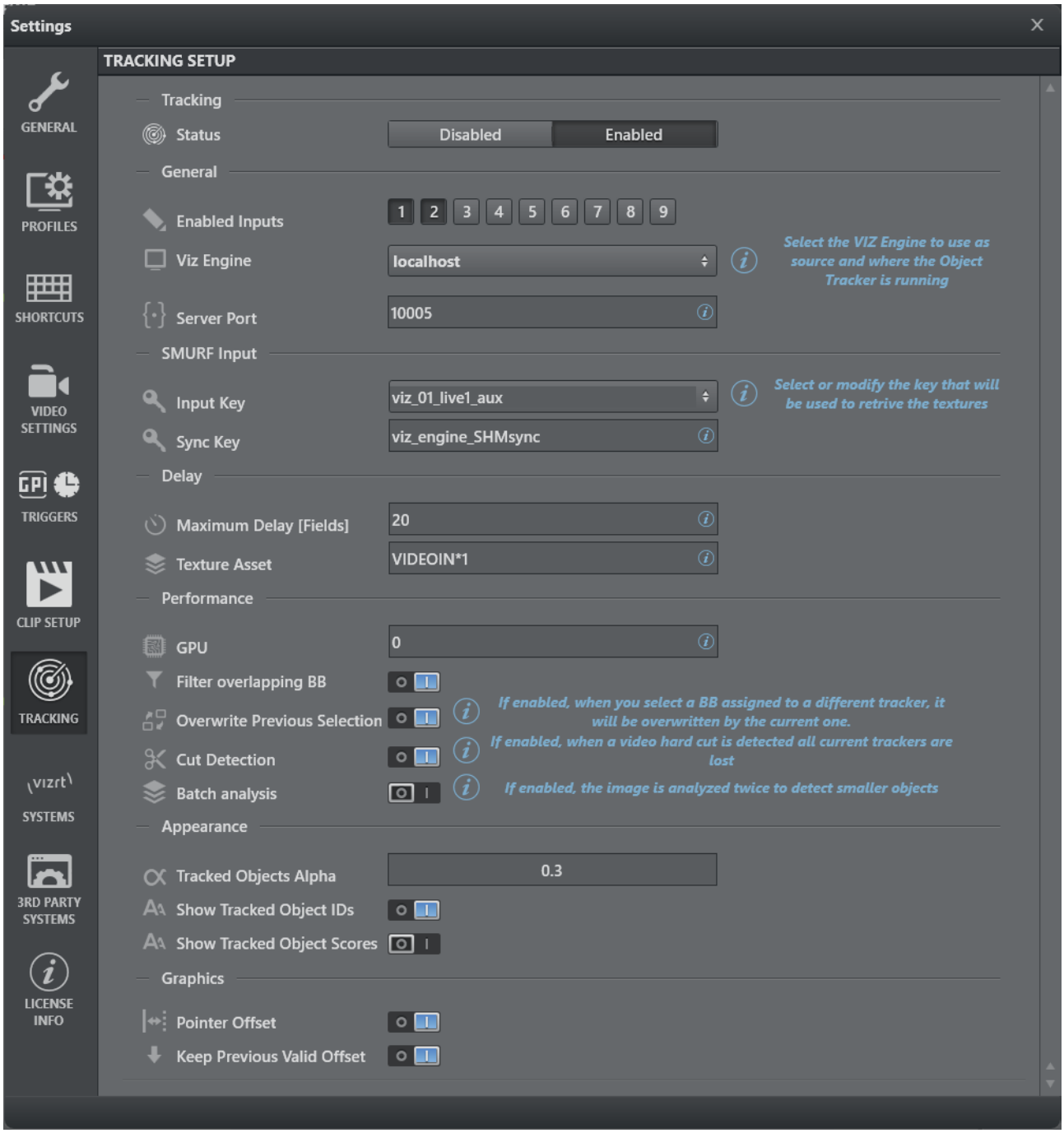
- Viz Arc license details.
- General Software Copyright and licensing.

Info: Contact your local Vizrt Support Representative or send an email to license@vizrt.com to obtain a license.

Copyright © 2025 Vizrt

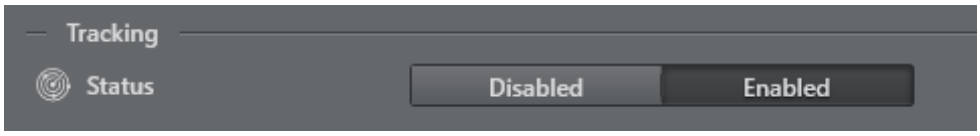

Settings
GENERAL
PROFILES
SHORTCUTS
VIDEO SETTINGS
GPI SETUP
CLIP SETUP
TRACKING
SYSTEMS
3RD PARTY SYSTEMS
INFO
LICENSE INFORMATION
License expires on 31/01/2023
Type: CORE Serial: 3-3695087
COPYRIGHT AND LICENSE DETAILS
Copyright (c) 2016-2022 vizrt. All rights reserved.
No part of this software, documentation 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.
Legal Notices.
This vizrt software may utilize the following copyrighted material.
AvalonEdit, ExcelDataReader, ExcelDataReader.DataSet, Gmap.NET, HtmlAgilityPack, ICSharpCode.WpfDesigner, MathNet.Numerics, MathNet.Spatial, Melanchall.DryWetMidi, MIConvexHull, Microsoft.ClearScript, Microsoft.ClearScript.V8.ICUData, MQTTnet.AspNetCore, MQTTnet.Extensions.ManagedClient, MQTTnet.Extensions.WebSocket4Net, Newtonsoft.Json, StreamDeckSharp and OpenMacroBoard.SDK
are distributed under the MIT License, which is displayed below:
MIT License
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4.10 Tracking



Copyright © 2025 Vizrt

4.10.1 Tracking



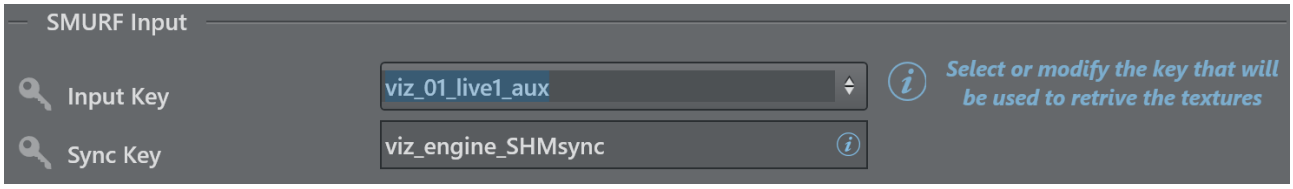
- **Status:** Enables or disables the Tracking panel.

4.10.2 General



- **Enabled Inputs:** Sets the number of objects that can be tracked contemporarily.
- **Viz Engine:** Select the Viz Engine, already pre-configured in the Profiles tab, that the Object Tracker uses as source.
- **Server Port:** Sets the port where the Viz Object Tracker is listening.

4.10.3 SMURF Input



- **Input Key:** Select (from a list) or modify the key to use to retrieve the textures for the Object Tracker.
- **Sync Key:** Viz Engine's SHM master sync key. The default is *viz_engine_SHMsync.*

4.10.4 Tracking Hub

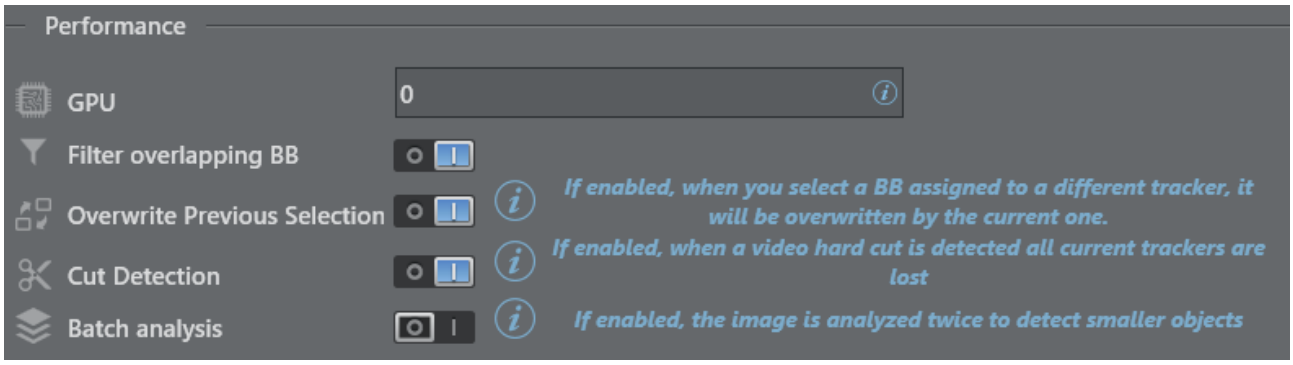


- **Maximum Delay |Fields|:** Texture delay set in the Engine, the delta fields between this value and the processing time is handled by the Object Tracker.

Copyright © 2025 Vizrt

- **Texture Asset:** Texture to delay; this parameter is automatically filled when the Input Key is selected from the list.

4.10.5 Performance



- **GPU:** Zero-based index representing GPU affinity where the Viz Object Tracker does parts of the AI computation. This applies only to multi GPU systems. `0` represents the fist GPU, `1` the second etc.
- **Filter overlapping BB:** Ignores bounding boxed that overlap to a certain extend with a larger bounding box.
- **Overwrite Previous Tracking:** Overwrites the currently selected tracker when selecting a bounding box that is already assigned to a different tracker, if enabled.
- **Cut Detection:** When a hard cut is detected, all current trackers are lost.
- **Batch analysis:** The image is analyzed twice to detect smaller objects, adding more stress both on CPU and GPU.

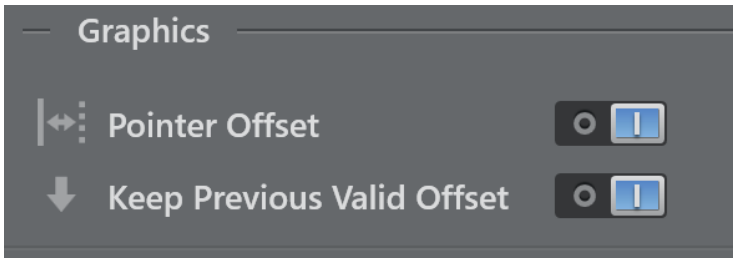
4.10.6 Appearance



- **Tracked Objects Alpha:** Sets the transparency of the bounding boxes of the detected objects.
- **Show Tracked Objects IDs:** Hides/shows the objects IDs when the detection is running, for debug purposes.
- **Show Tracked Objects Scores:** Hides/shows the objects scores when a detection (Detection and Tracking, Face and Pose) is running, for debug purposes.

Copyright © 2025 Vizrt

4.10.7 Graphics

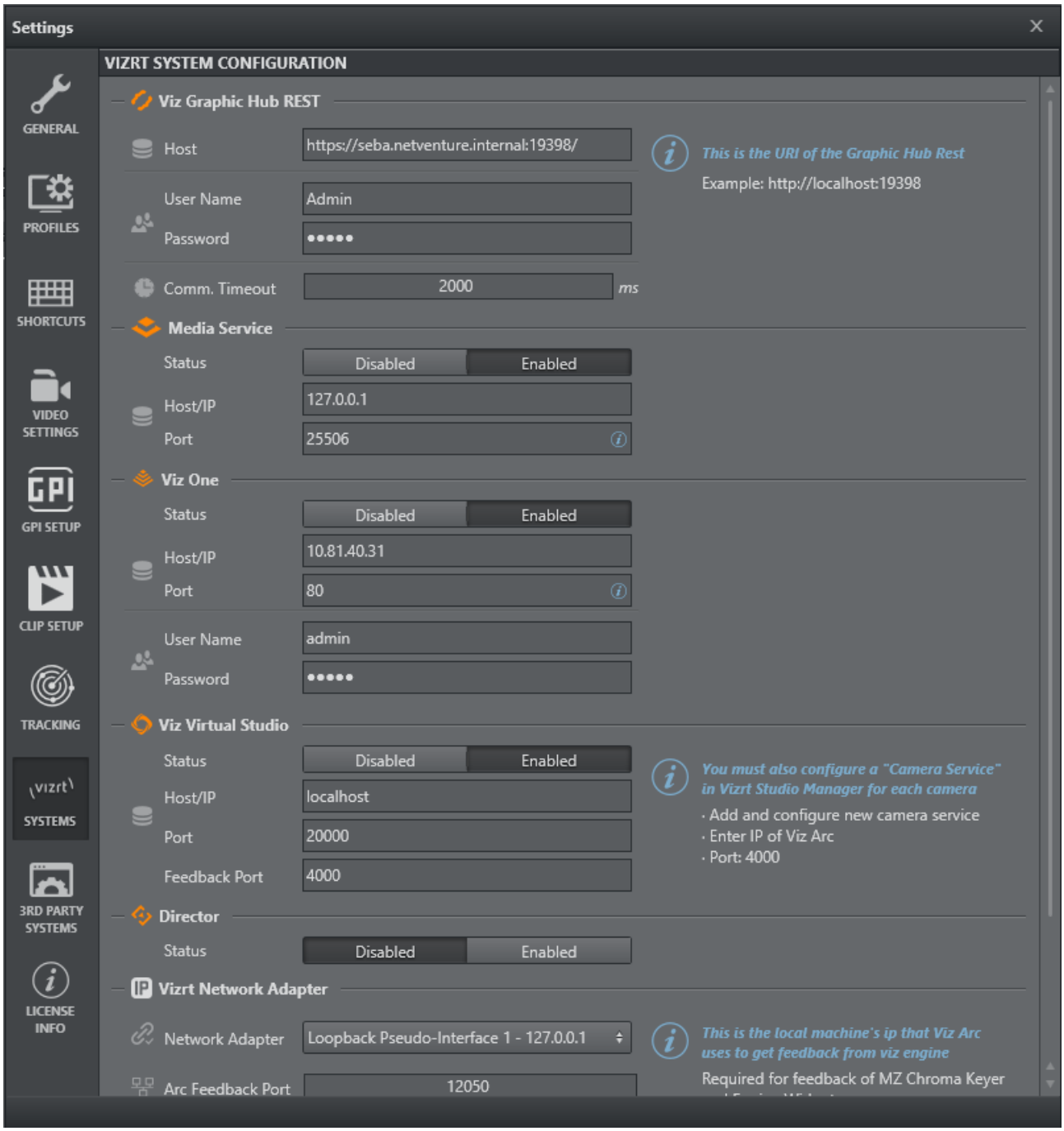


- **Pointer Offset:** Enables the possibility to draw an offset to apply to the pointer.
- **Keep Previous Valid Offset:** Applies the the last valid offset each time you click on a new object when active. Otherwise, the offset you draw each time is applied.

Copyright © 2025 Vizrt

4.11 Vizrt Systems

This part of the configuration groups all Vizrt related systems that connect to Viz Arc.

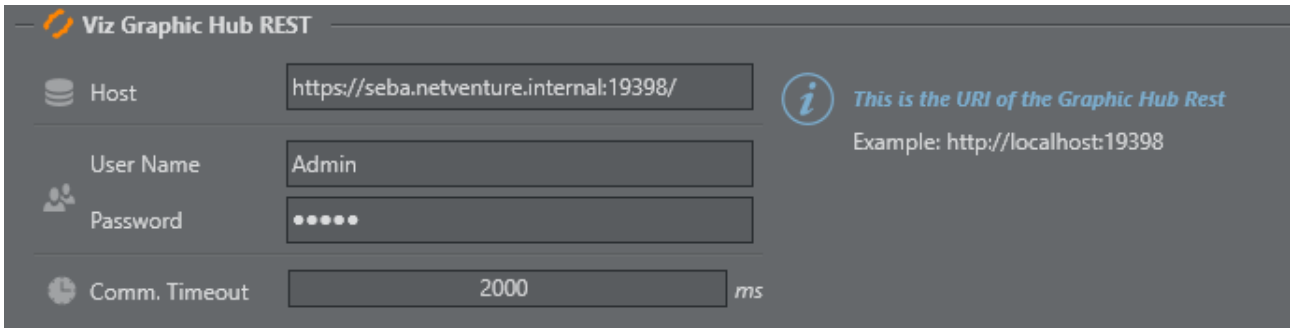


4.11.1 Graphic Hub REST

Viz Arc uses the Graphic Hub REST API for browsing scenes, images, merged geometries and materials.

Copyright © 2025 Vizrt

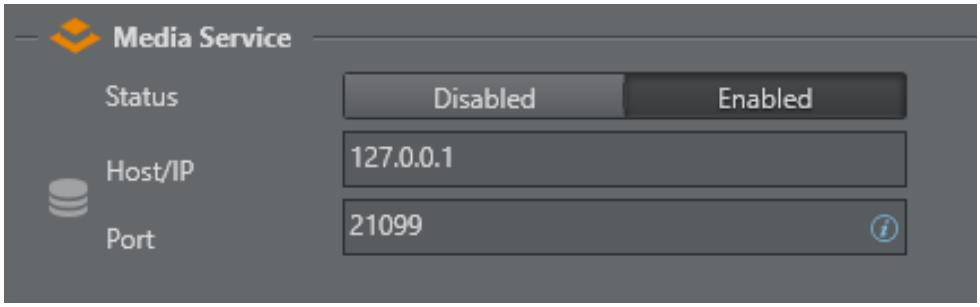
The following settings related to Graphic Hub can be configured:



- **SSL:** Switches between HTTP or HTTPS connection to the Graphic Hub REST.
- **Host/IP:** Host or IP of the Graphic Hub REST service.
- **Port:** Connection port to be utilized. The default port is `19398`.
- **User Name & Password:** Login credentials for Graphic Hub.
- **Comm. Timeout:** The timeout in milliseconds for a REST request.

4.11.2 Media Service

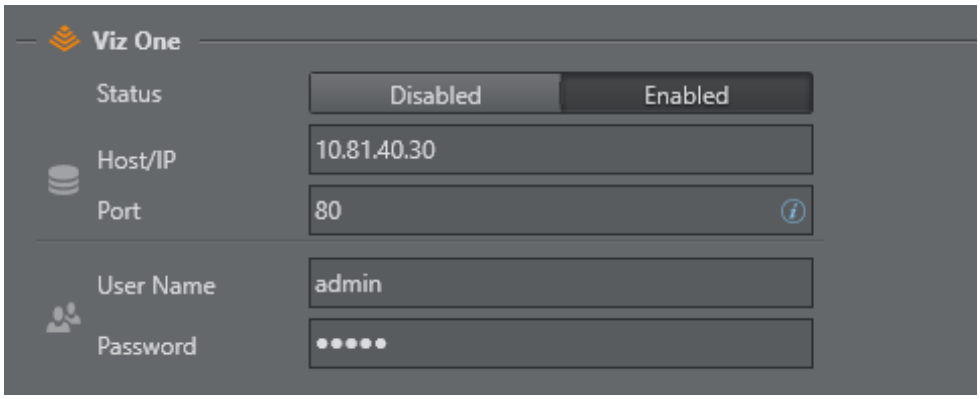
Viz Arc uses Media Service for asset browsing images and clips. Media Service can also be used to record the Engine's output.



- **Status:** Enables or disables the Media Service.
- **Host/IP:** Host or IP where Media Service is installed.
- **Port:** Connection port to be utilized. The default port is `21099`.

Copyright © 2025 Vizrt

4.11.3 Viz One



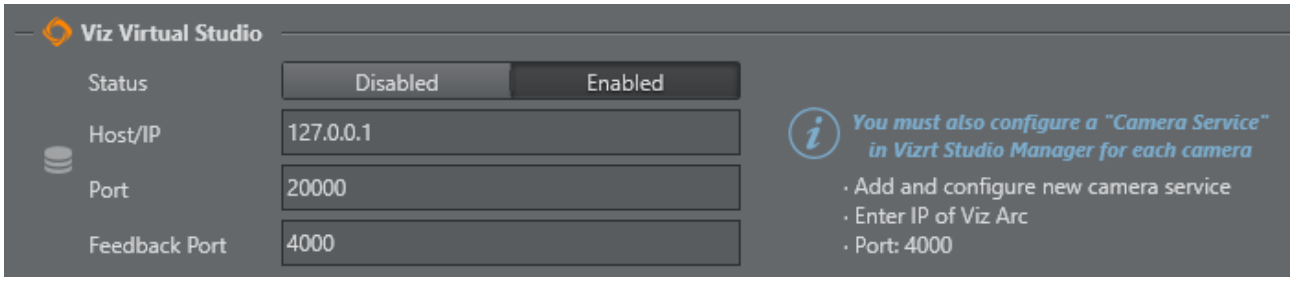
- **Host/IP:** Host or IP where Viz One server is installed.
- **Port:** Connection port to be utilized. The default port is `80`.
- **User Name & Password:** Login credentials for Viz One (default admin/admin).

The Viz One server can be used as source of image assets for **Image Actions** or as source of video clips for the **Clip Action.**

4.11.4 Viz Virtual Studio

When working with tracked cameras in a Virtual Studio environment, Viz Arc communicates with Tracking Hub or VizIO to send and receive camera tracking information.

In order to receive tracking information, you must select the correct **Tracking Source.** The following settings related to Viz Virtual Studio can be configured:



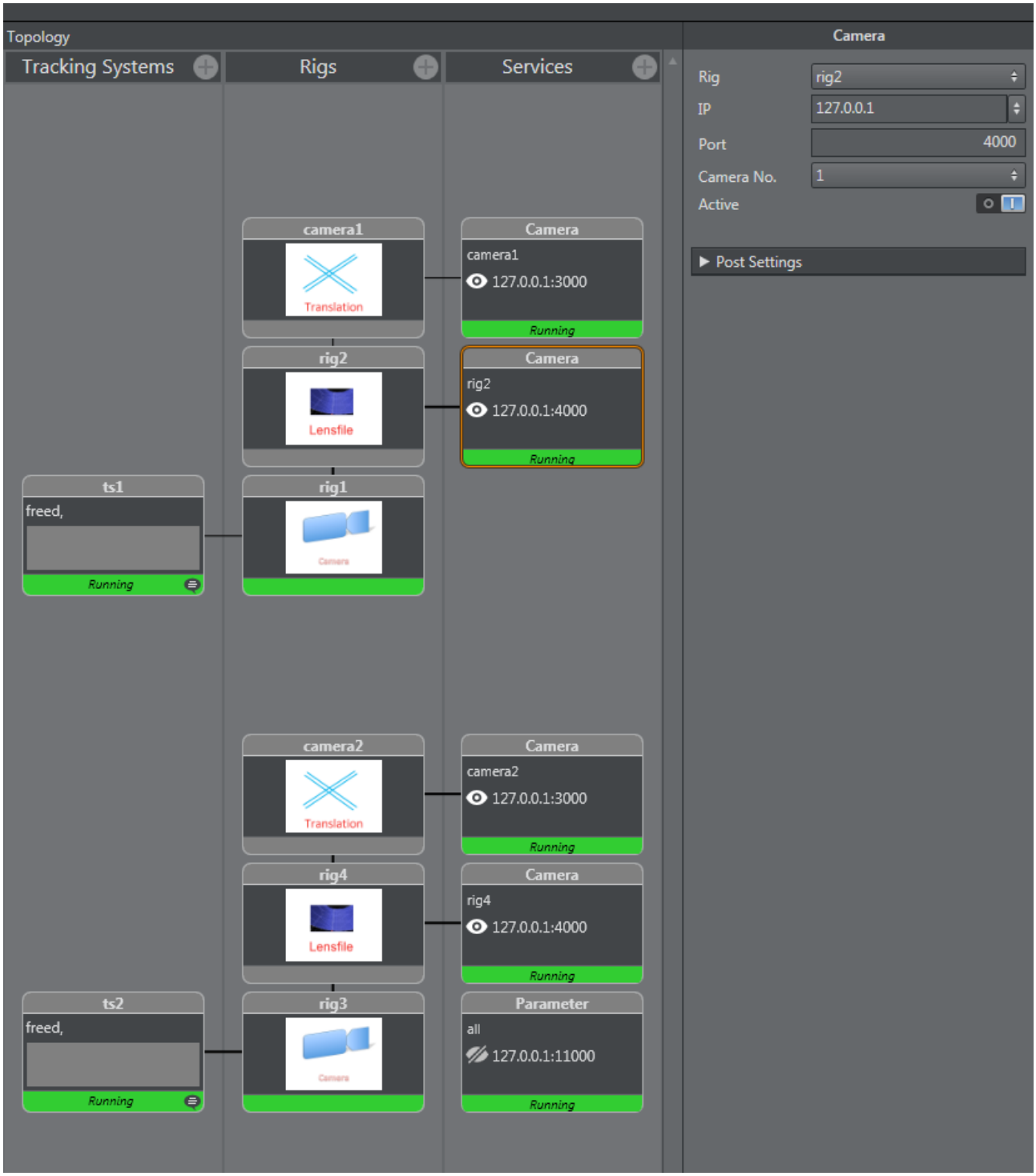
- **Status:** Enables or disables the Virtual Studio.
- **Host/IP:** Host or IP of the Tracking Hub.
- **Port:** Connection port to be utilized. The default port for Tracking Hub is `2000.`
- **Feedback Port:** The port on which Viz Arc listens for camera updates from the Tracking Hub.

Important! When working with Tracking Hub, you must configure a **Camera Service** in Viz Studio Manager for each camera:

- Add and configure a new camera service.
- Enter IP of Viz Arc.

Copyright © 2025 Vizrt

- Port must match the port number specified in Feedback Port (default `4000`).



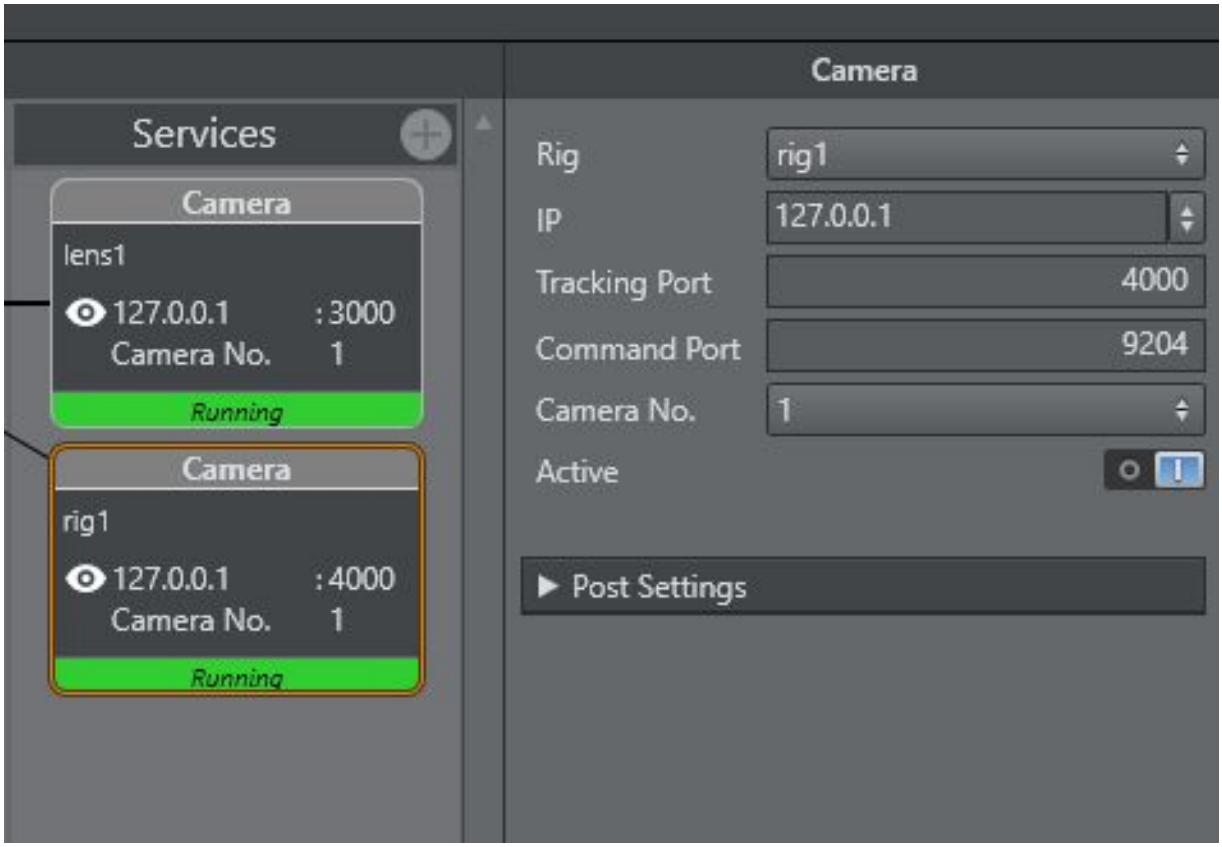
Copyright © 2025 Vizrt

Configuring Rigs to Viz Engine and Viz Arc

In Tracking Hub, a parent translation rig cameraX (where X is the number of the tracked camera) must be created. This rig must be sent to the Viz Engine. The rig below, which contains the lens information, must be sent to Viz Arc on the port specified in **Feedback Port** (default `4000`).

> **Note:** The configuration above makes it possible to move the camera on the maps section. Viz Arc sends the camera position to the translation rig cameraX when moving the camera along the map. This kind of setup is NOT mandatory and depends on the workflow.

4.11.5 TCP Command Port



Tracking Hub might output some connection errors on the console when adding the **Camera Service** for Viz Arc**.** To remove those error messages, configure the **Command Port** to match the configured TCP communication port (default `9204`).

4.11.6 Director

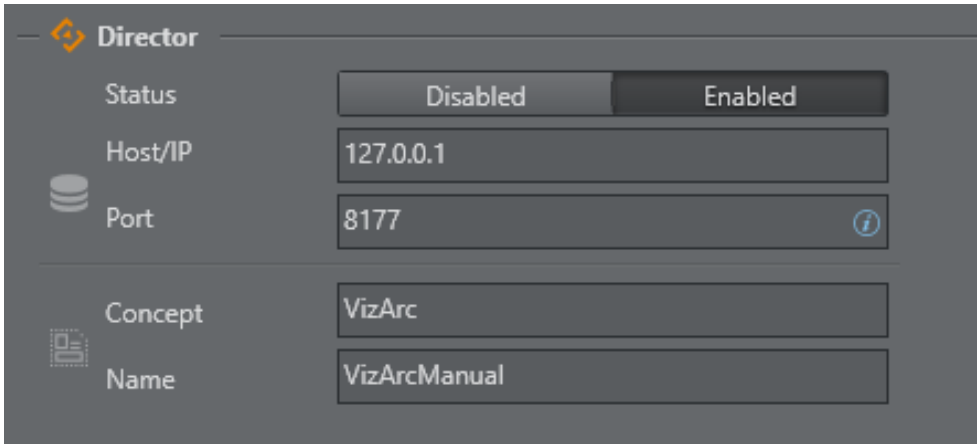
Viz Arc can be connected to Viz Pilot, which is a program used to send graphics On Air to Viz Engine.

Copyright © 2025 Vizrt

Note: These settings are related to Pilot Data Server, an application that acts as a server for accessing the Director database and other services.

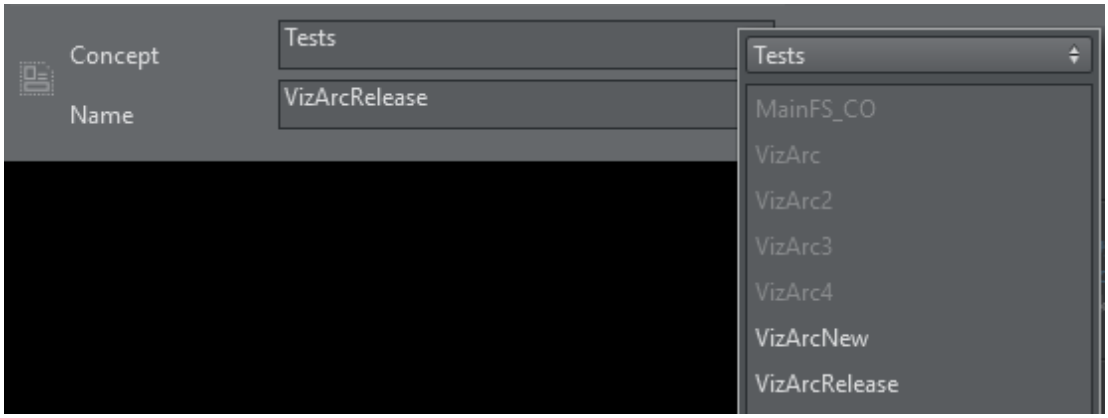
For more information about Viz Pilot, see the Viz Pilot User Guide.

The following settings related to Pilot Data Server can be configured in the menu below:



- **Host/IP:** Host or IP of Pilot Data Server.
- **Port:** The default port is `8177`.

It's also possible to select the **Concept** and the template using the edit icon:



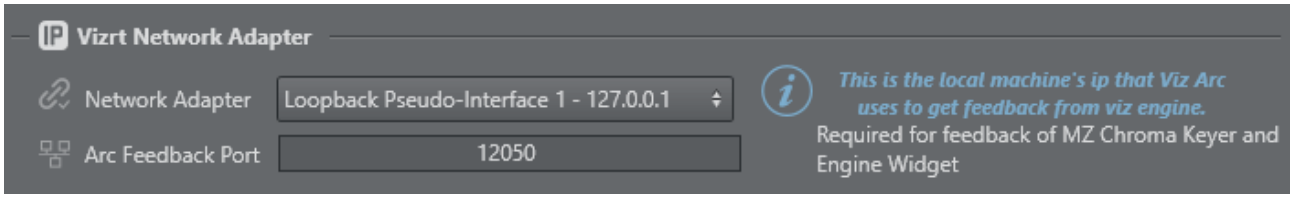
The following must be installed in order to use the interaction with Viz Director:

- Viz Arc REST Service
- Viz Pilot Data Server

Vizrt Network Adapter

The selected Network Adapter is used to listen to feedback from the Viz Engines. This feedback is used to get information about chroma key settings and Engine status used in the Engine Status Widget.

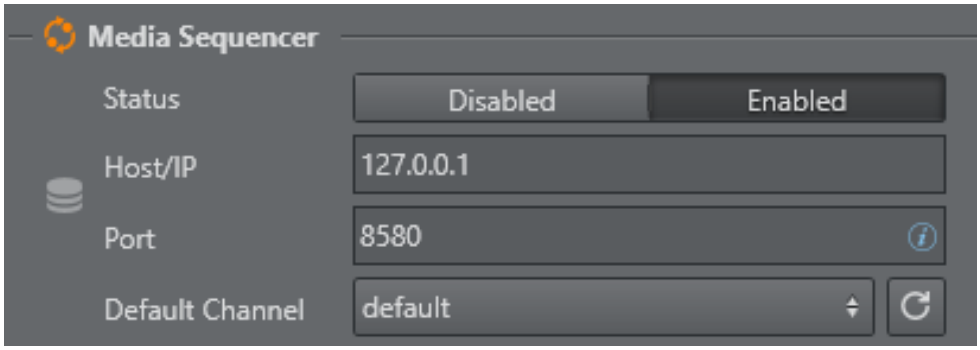
Copyright © 2025 Vizrt



- **Network Adapter:** The network adapter to listen on. Make sure you select the network adapter the Vizrt network is configured on.
- **Arc Feedback Port:** The Port on which Viz Arc listens for UDP feedback from the Viz Engines.

Media Sequencer

Configure the Media Sequencer to use MSE actions to **take/continue/take out** Viz Trio pages using MSE's REST service.



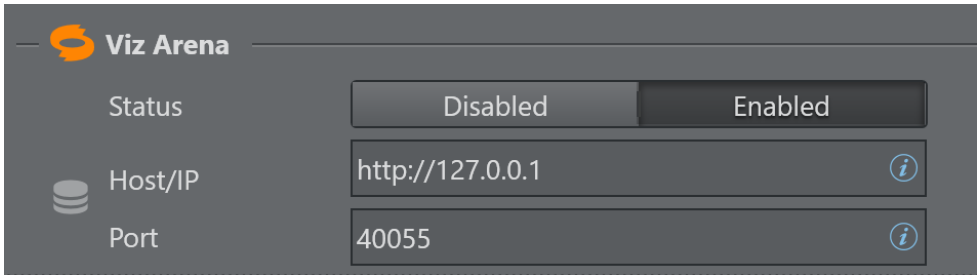
Viz Arc uses Media Sequencer (MSE) to connect to Viz Engine. Through MSE you can also retrieve saved pages in Viz Trio using the Media Sequencer REST API and import them directly into Viz Arc for execution.

The following settings related to MSE can be configured in the menu above:

- **Host/IP:** Host or IP of MSE.
- **Port:** The default port of the MSE REST service is `8580`.
- **Default Channel:** The default channel to be used on the creation of a MSE action. Use the refresh button to re-fetch a list of available output channels. The selected channel is used as default output channel when a MSE action is created.

Viz Arena

Configure the Viz Arena server for operating Viz Arena through Viz Arc.



- **Status:** Enables or disables the Viz Arena integration.

Copyright © 2025 Vizrt

- **Host/IP:** Host or IP where Viz Arena server is running.
- **Port:** Connection port to be utilized. The default port is `40055`.

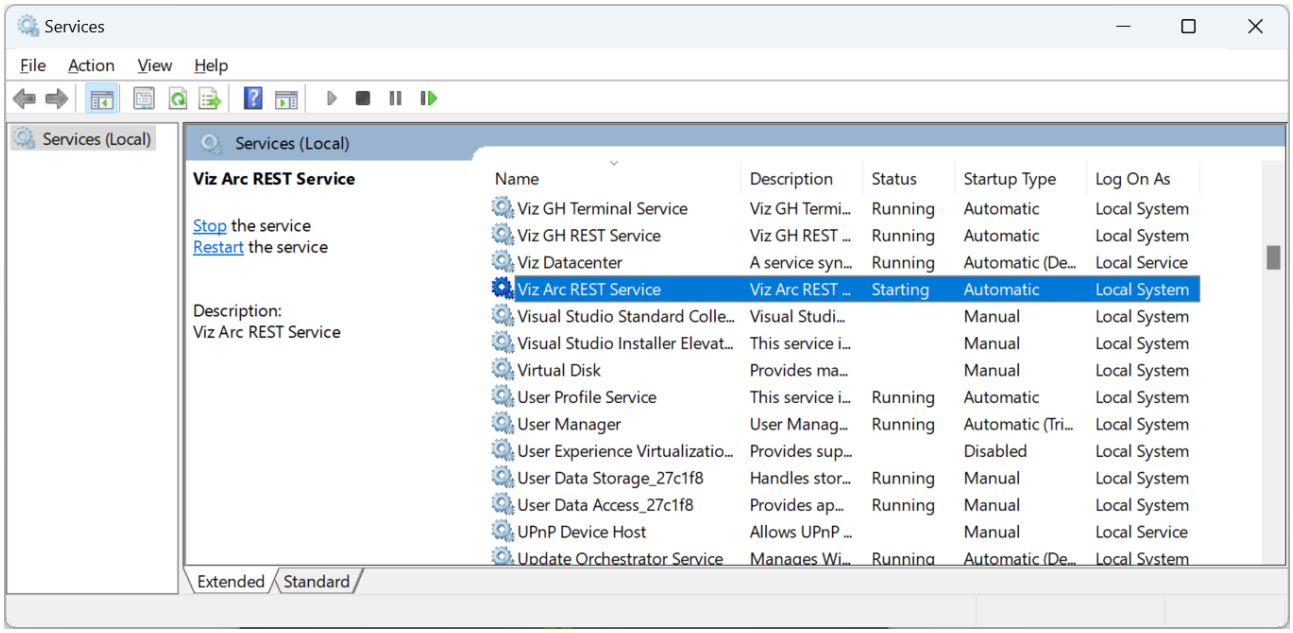
Viz Arc REST Service

Install the Viz Arc REST service by checking the **Viz Arc REST Service** option in the bundle installer.



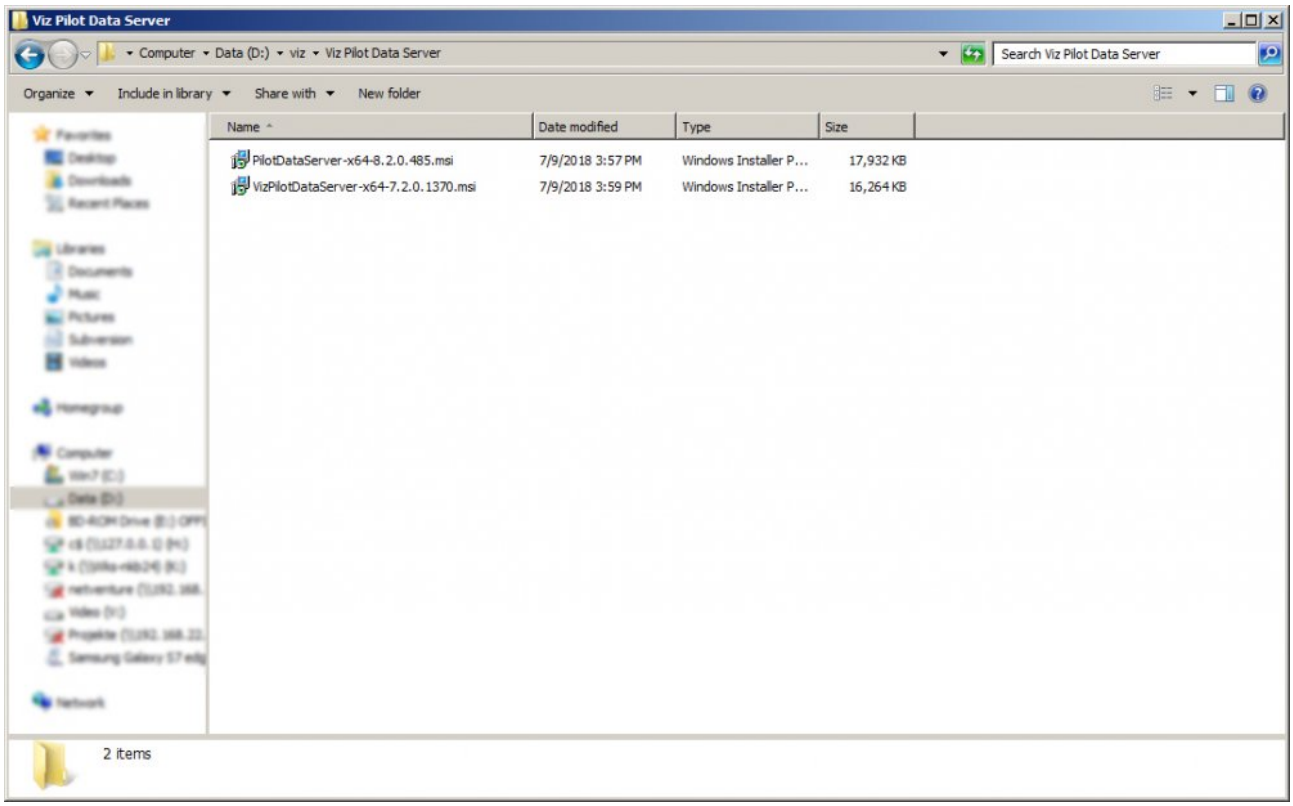
Once installed, Viz Arc REST Service, you can check if the service is running:

Copyright © 2025 Vizrt



License: The Viz Arc Web Service requires the **Viz Arc REST** license to be in place, otherwise the service will not start.

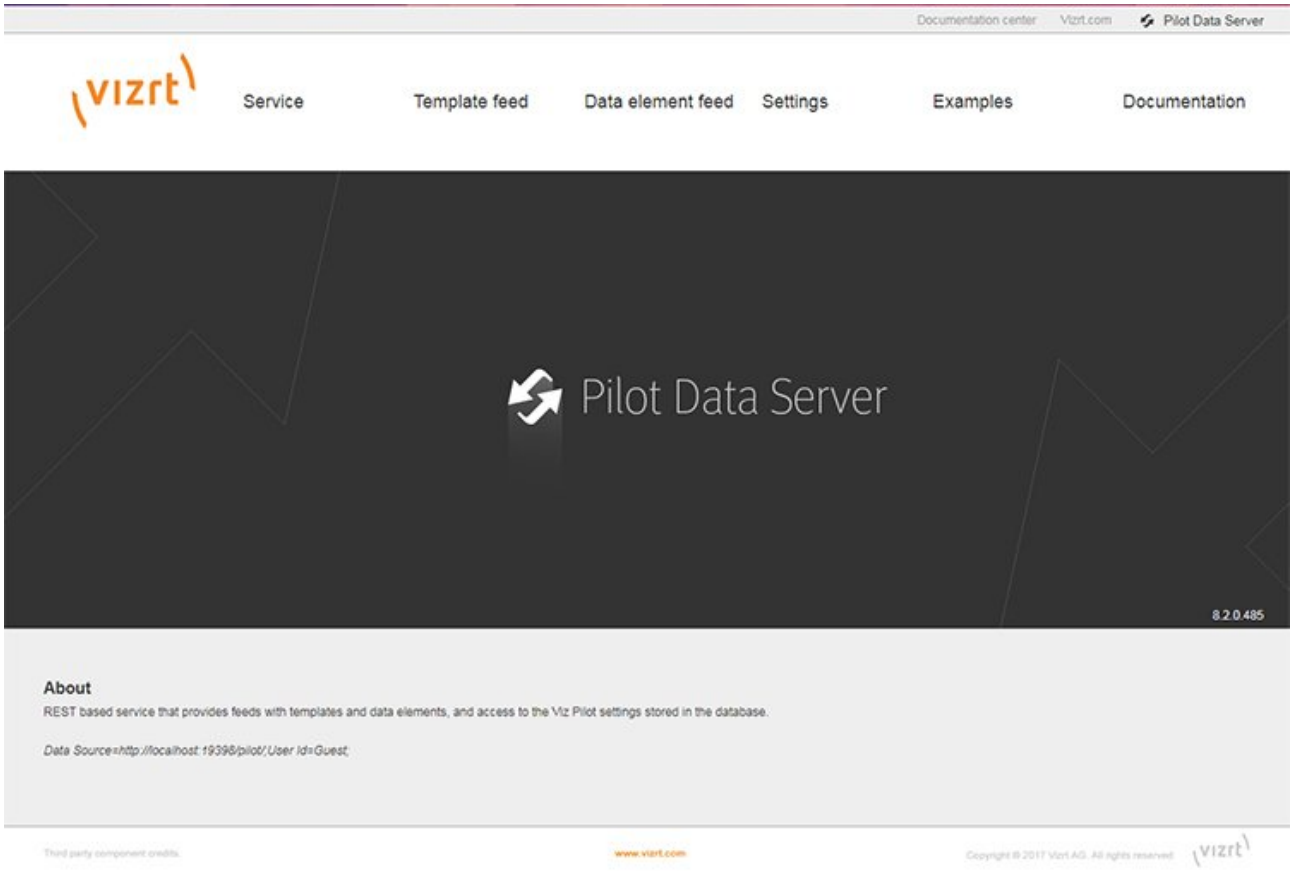
Viz Pilot Data Server



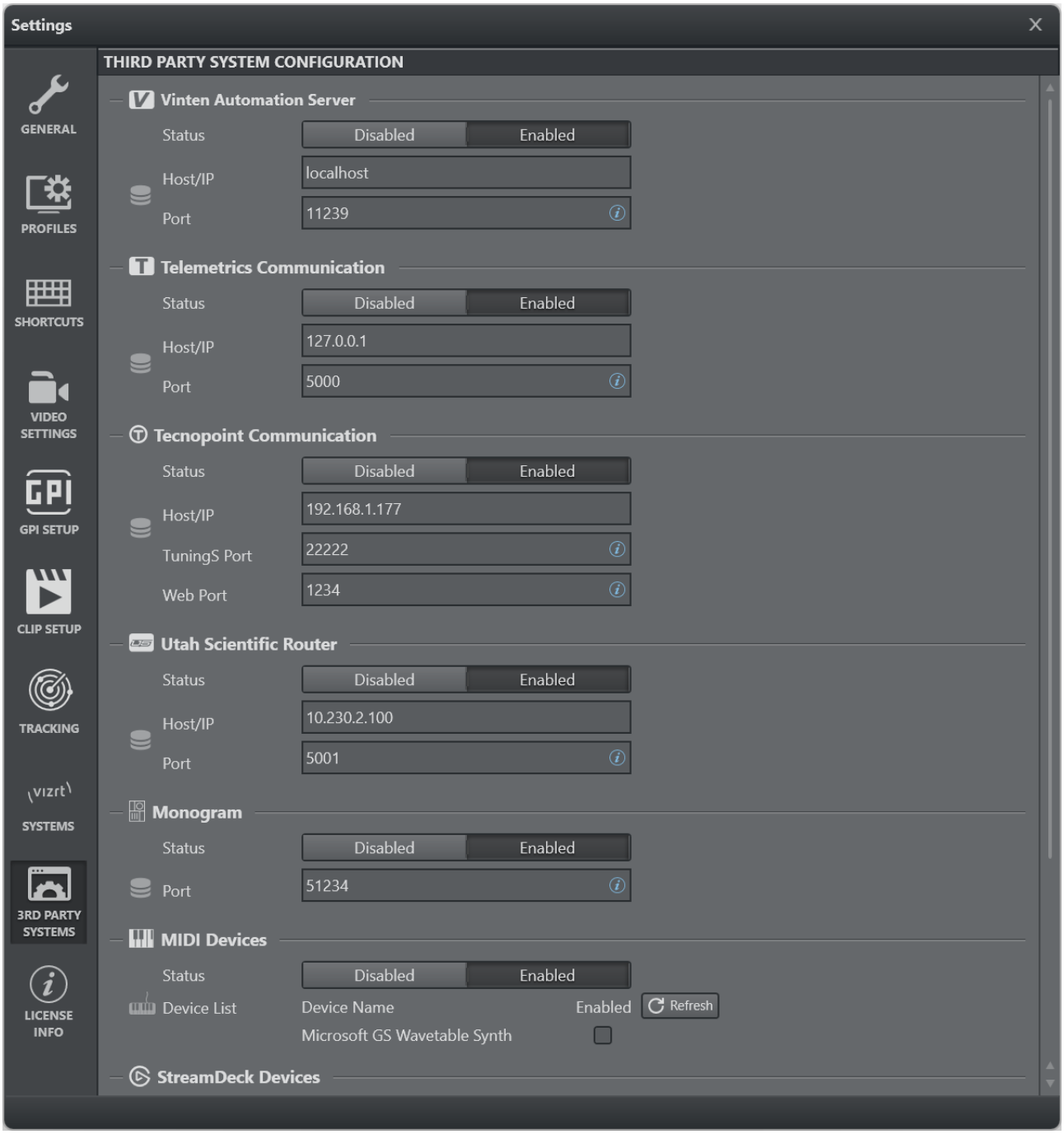
Copyright © 2025 Vizrt

Open the Pilot Data Server installer and follow the instructions.

When you have finished installation, you can test if the server is up and running by entering the IP and port (default is `8177`) in the browser's address bar. You should see this page:



4.12 Third Party Systems



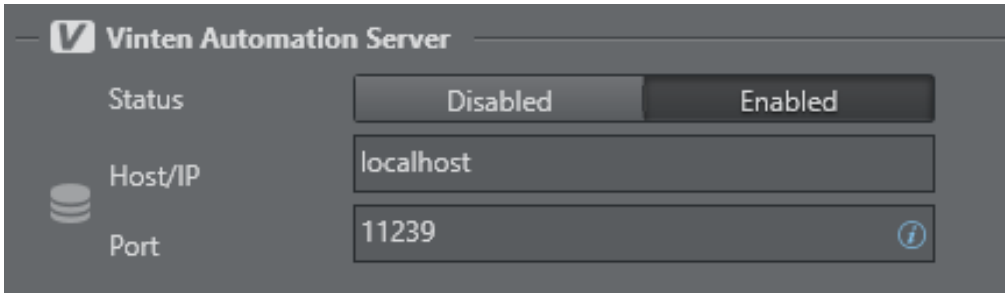
- Vinten Automation Server
- Tecnopoint Automation Server
- Telemetrics Automation Server
- Utah Scientific Router
- Monogram

Copyright © 2025 Vizrt

- MIDI Device
- Stream Deck
- Art-Net DMX

4.12.1 Vinten Automation Server

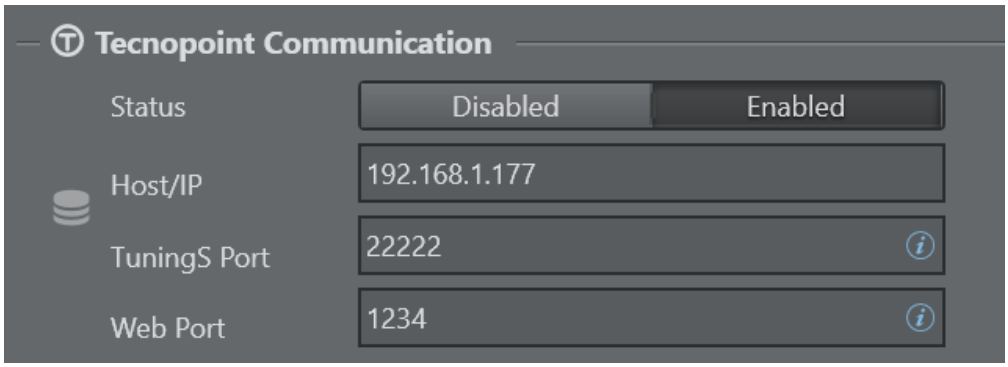
Viz Arc can be connected to a Vinten Automation Server. Vinten is a supplier of manual and robotic camera support systems.



- **Status:** Enables or disables the server.
- **Host/IP:** Host or IP of where the Vinten server is installed.
- **Port:** The default port is `11239`.

4.12.2 Tecnopoint Automation Server

Viz Arc can be connected to a Tecnopoint Automation Server.

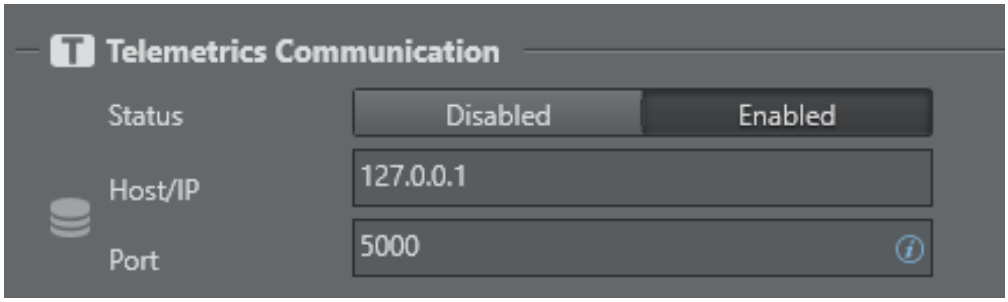


- **Status:** Enables or disables the server.
- **Host/IP:** Host or IP of where the Tecopoint TuningS server is installed.
- **TuningS Port:** The default port is `22222`.
- **Web Port:** The HTTP port used to fetch thumbnail images, the default is `1234`.

4.12.3 Telemetrics Automation Server

Viz Arc can be connected to a Telemetrics Automation Server.

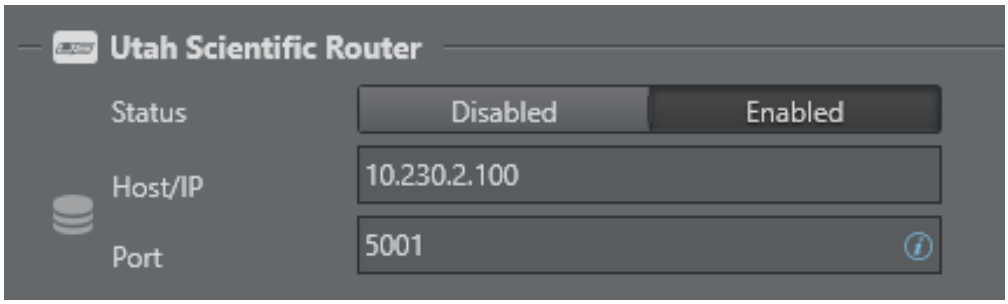
Copyright © 2025 Vizrt



- **Status:** Enables or disables the server.
- **Host/IP:** Host or IP of where the Telemetrics server is installed.
- **Port:** The default port is `5000`.

4.12.4 Utah Scientific Router

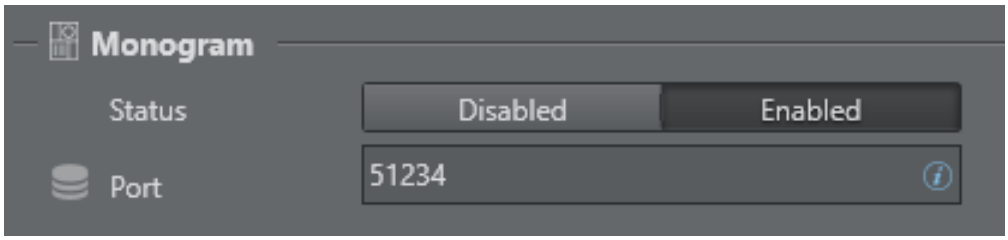
Viz Arc can connect to a Utah Scientific Video Router.



- **Status:** Enables or disables the router.
- **Host/IP:** Host or IP of where the Utah router is installed.
- **Port:** The default port is `5001`.

4.12.5 Monogram

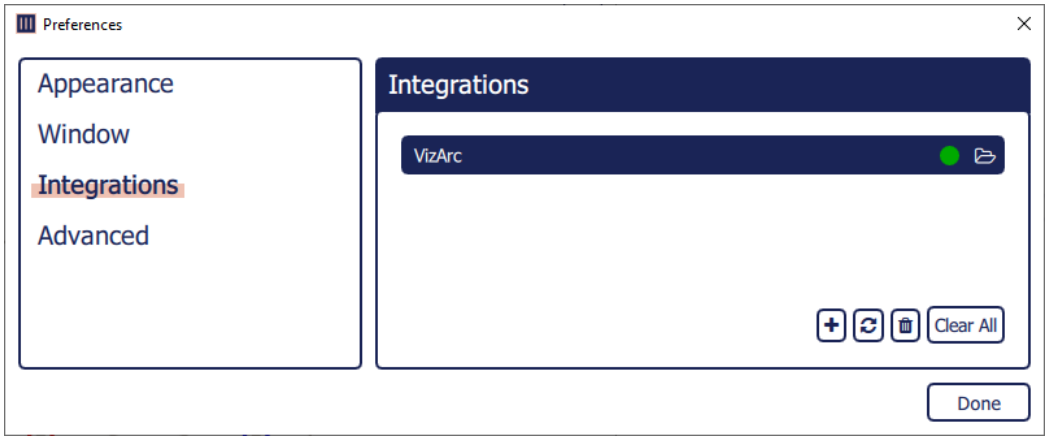
Viz Arc supports monogram consoles.



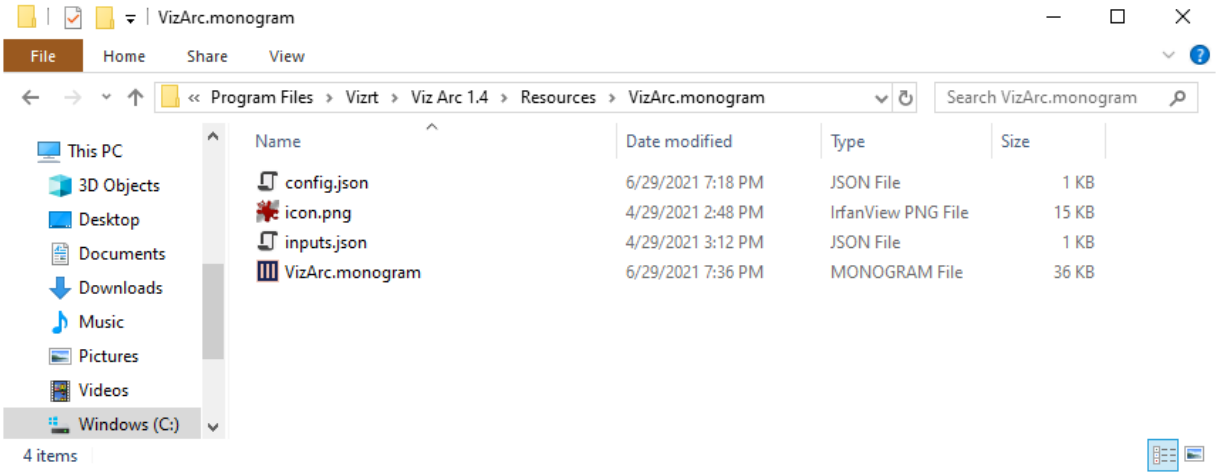
- **Status:** Enables or disables the console.
- **Port:** Sets the UDP port to listen for device input.

To configure monogram to work with Viz Arc, go to **Monogram Creator > Preferences** and add the Viz Arc integration.

Copyright © 2025 Vizrt



A predefined integration can be found in the Viz Arc program-data folder under **Resources/VizArc.monogram**.



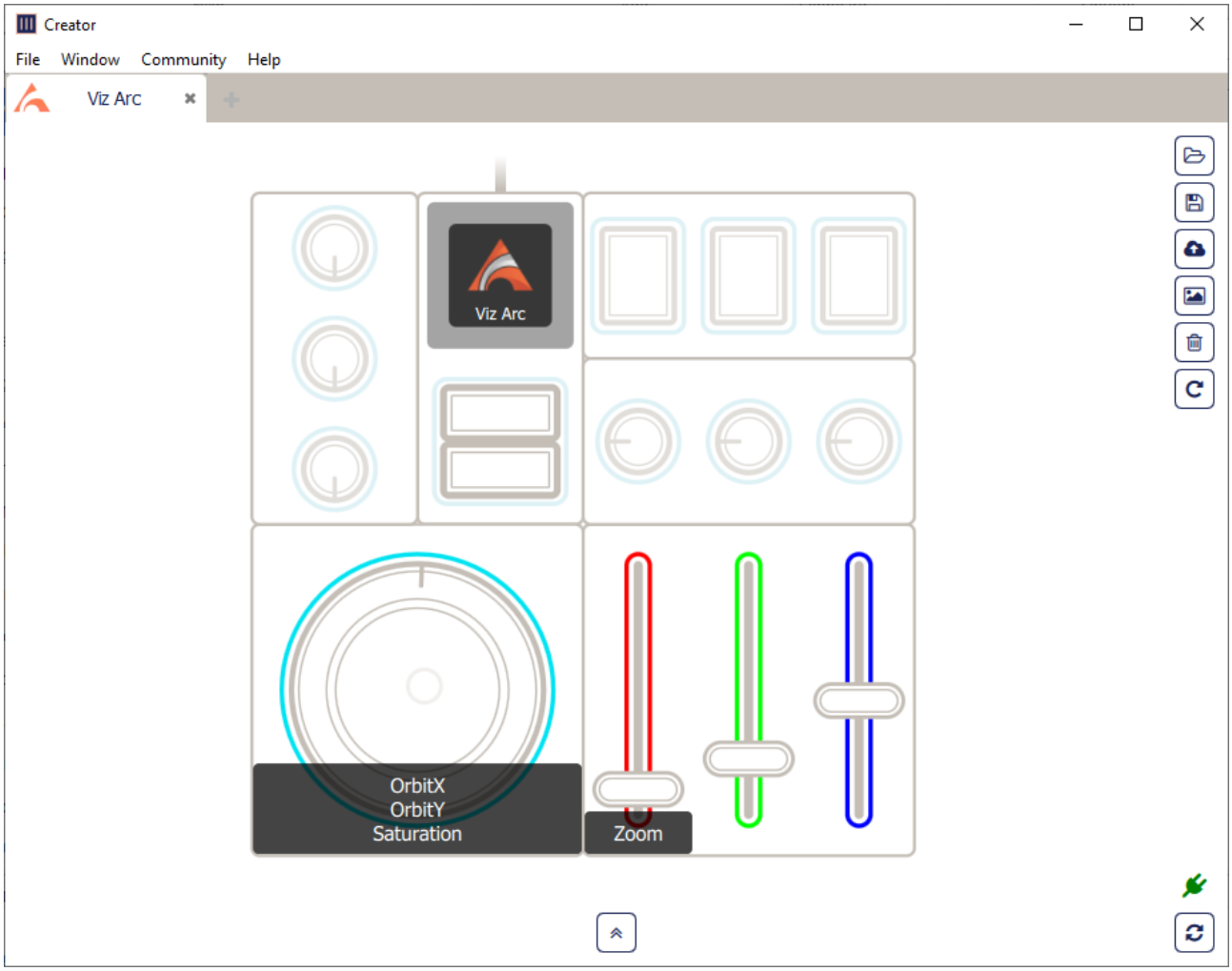
In the *config.json* fille, the **port** can be changed to something else than the `51234` default port.

config.json

```
{
  "name": "VizArc",
  "id": "vizrt.VizArc",
   "exe": [ "VizArc.exe" ],
  "connection": [
    {
      "name": "VizArcConnection",
      "type": "udp",
      "port": 51234
    }
  ]
}
```

When used with an **ORBITER** module, the integration can be used to control an NDI PTZ camera using the NDI Action.

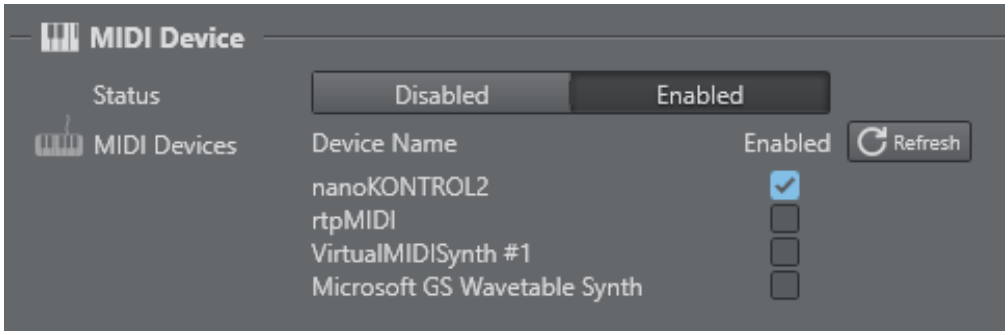
Copyright © 2025 Vizrt



A sample of connected monogram modules including an **ORBITER** module.

4.12.6 MIDI Device

Viz Arc can use one or more attached MIDI devices to control certain UI elements.



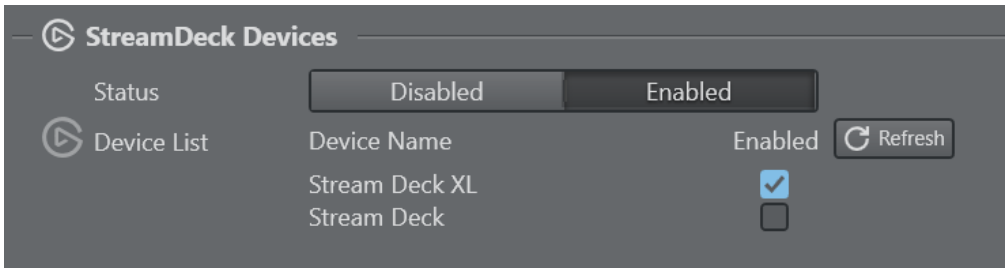
- **Status:** Enables or disables the MIDI devices.

Copyright © 2025 Vizrt

When the configuration window is opened or the refresh button is pressed, Viz Arc scans for connected MIDI devices and displays them as a list of available devices. Check **Enabled** if you want to use a device in Viz Arc. When checked, the device can be used through scripting identified by the **Device Name**.

4.12.7 Stream Deck

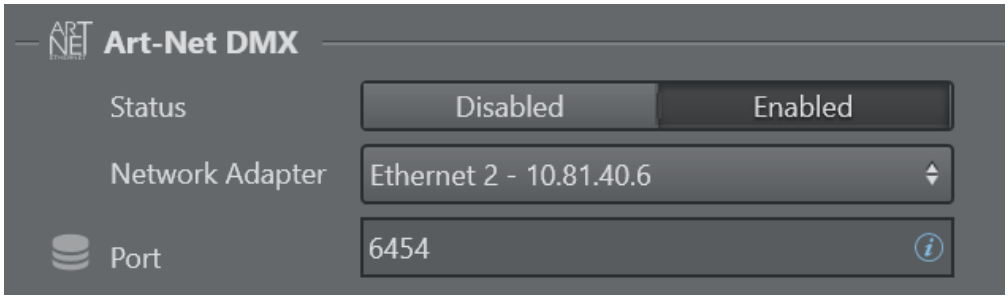
The **Stream Deck Devices** section lists all Stream Deck devices connected to the system.



- **Status:** Enables or disables Stream Deck device usage.
- **Device List:** Lists of all currently connected devices. Use the **Refresh** button to refresh the list in case a device has been connected or disconnected after starting Viz Arc. Uncheck any device you do not want to use in Viz Arc.

4.12.8 Art-Net DMX

The **Art-Net** section is used to configure the listening port and interface for Art-Net DMX lighting data.



- **Status:** Enables or disables **Art-Net** listening.
- **Network Adapter:** The network interface to listen on for Art-Net messages.
- **Port:** The port to use to listen to the messages, the default port is `6454`.

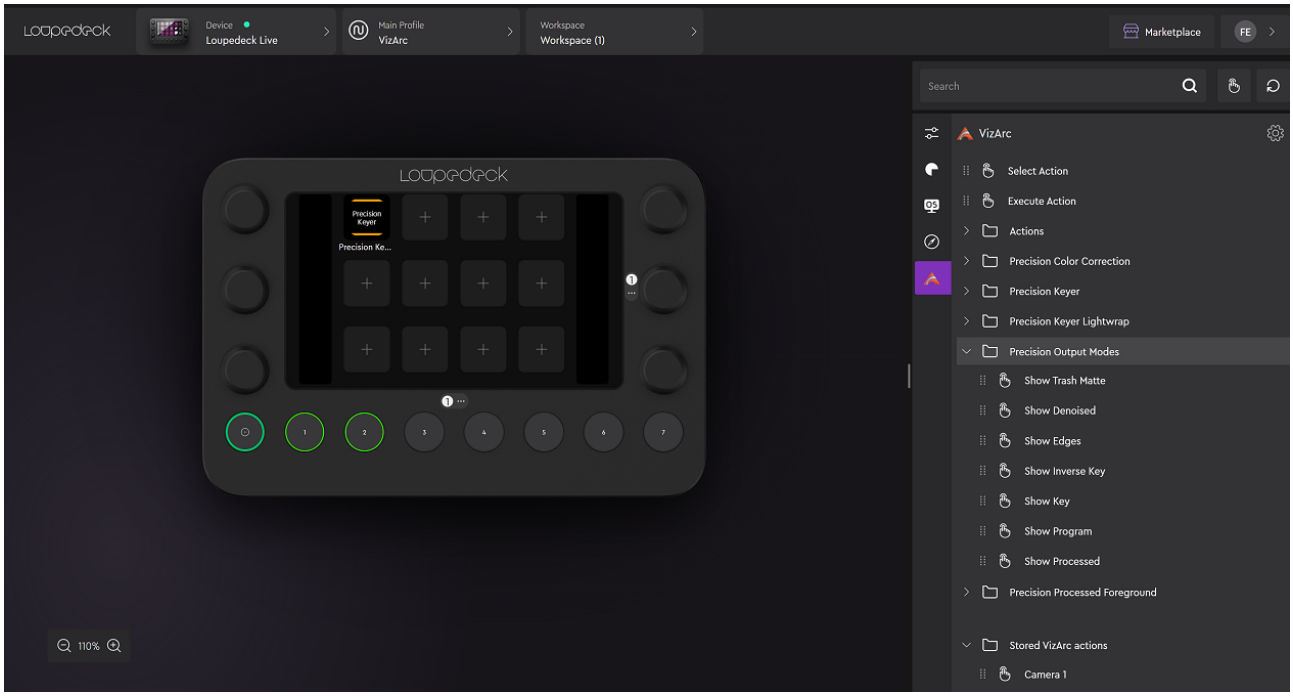
See Also

- Scripting Classes for more information on how to use MIDI devices.

Copyright © 2025 Vizrt

4.13 Loupedeck Integration

The Loupedeck device integration allows a user do adjust the Viz Engine Precision Keyer using the hardware dials and touch buttons. It also allows you to adjust the values of any transformation action allowing for quick size/ position/rotation changes of scene elements in addition to the ability to interact with the Precision Keyer action.

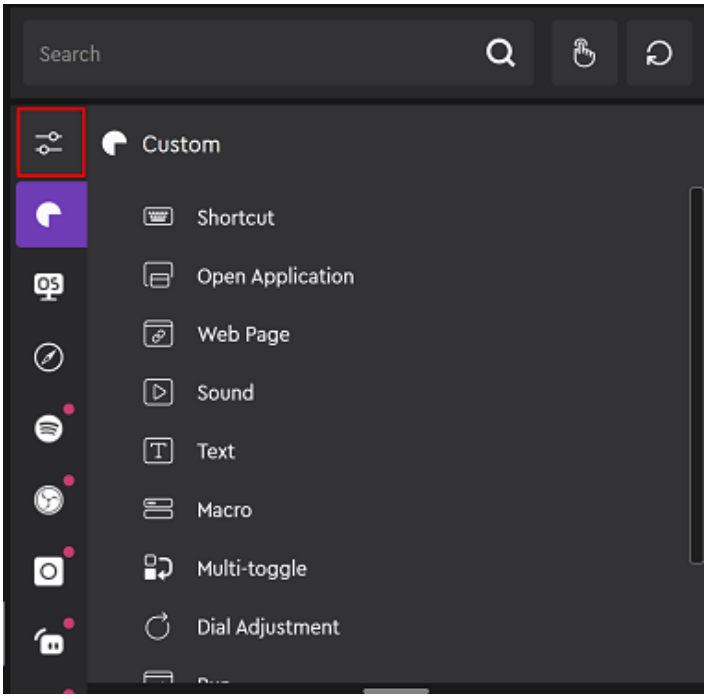


4.13.1 Installation

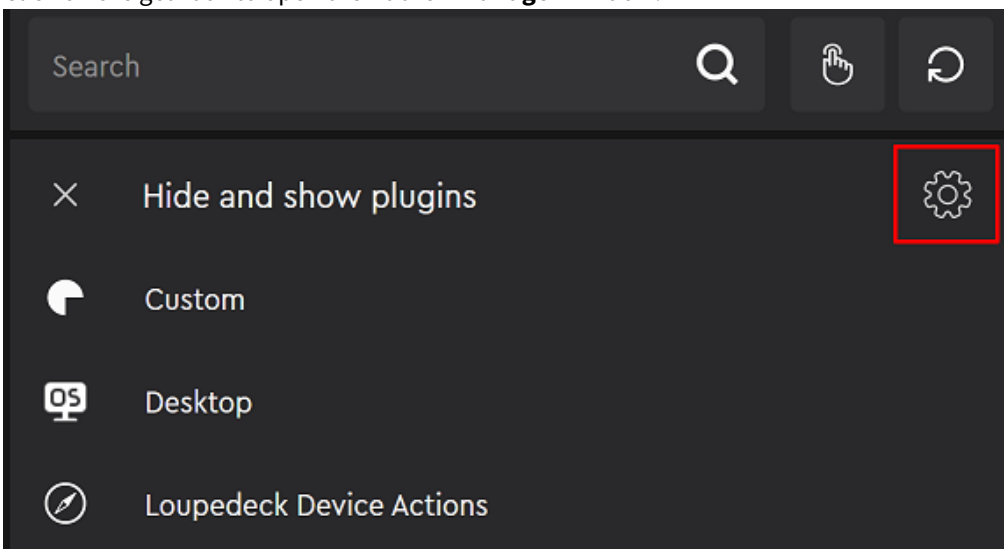
To use the Loupedeck integration, download the Loupedeck software from here. The software does **not** necessarily need to be installed on the same machine as Viz Arc.

- After installation, go to the Plugins Settings menu.

Copyright © 2025 Vizrt

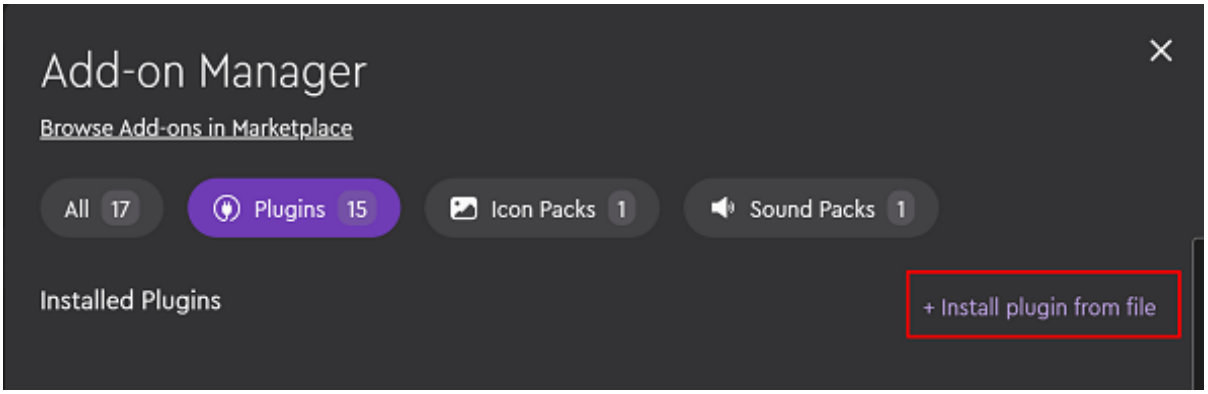


- Click on the gearbox to open the **Add-on Manager** window.

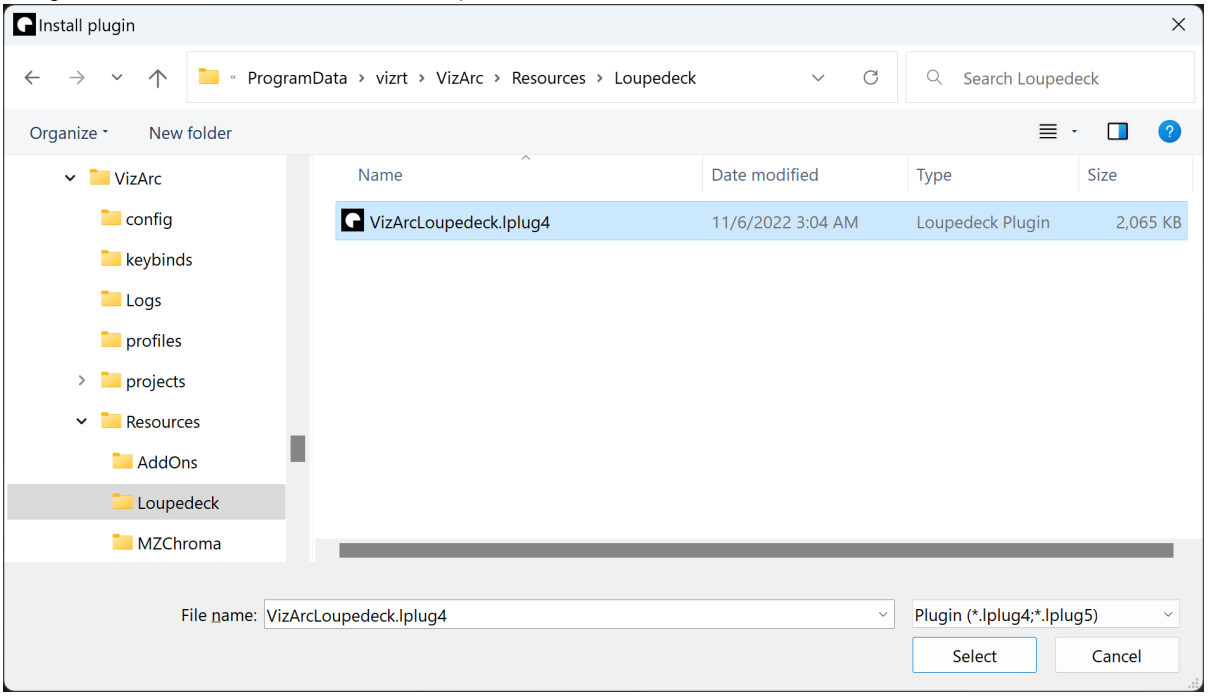


- Select **Install plugin from file**.

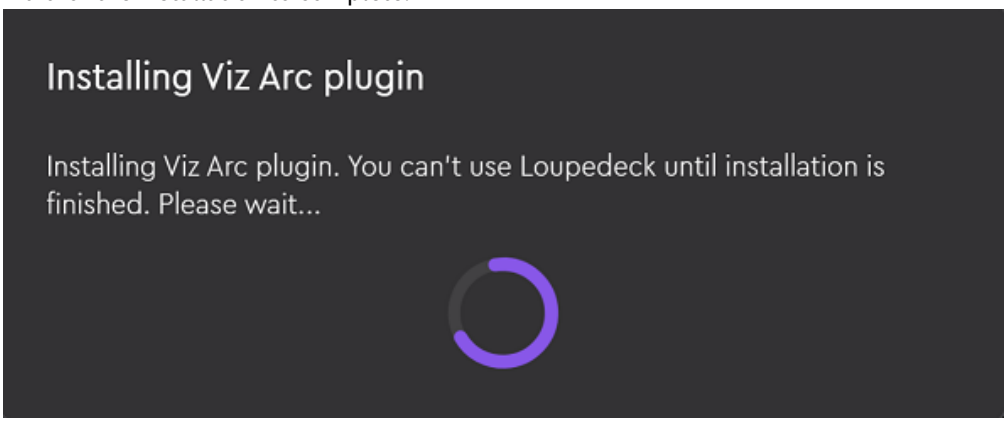
Copyright © 2025 Vizrt



- Browse for the Viz Arc plug-in contained in the file *VizArcLoupedeck.lplug4*, which can be found in *C:\ProgramData\vizrt\VizArc\Resources\Loupedeck* after Viz Arc installation.

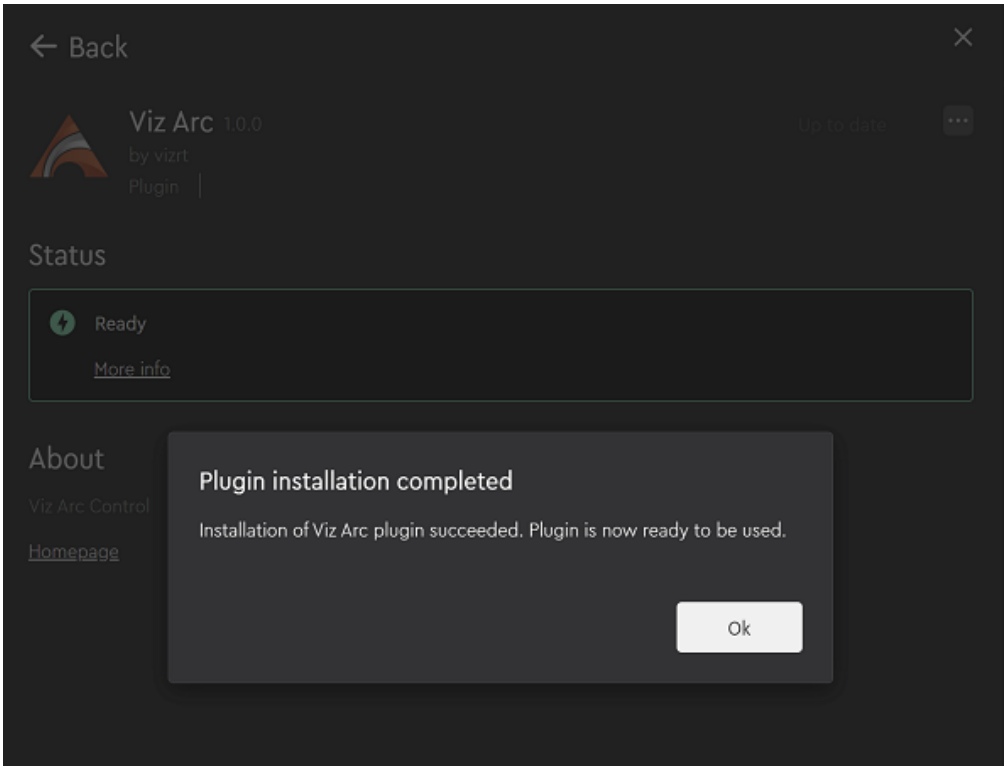


- Wait for the installation to complete.

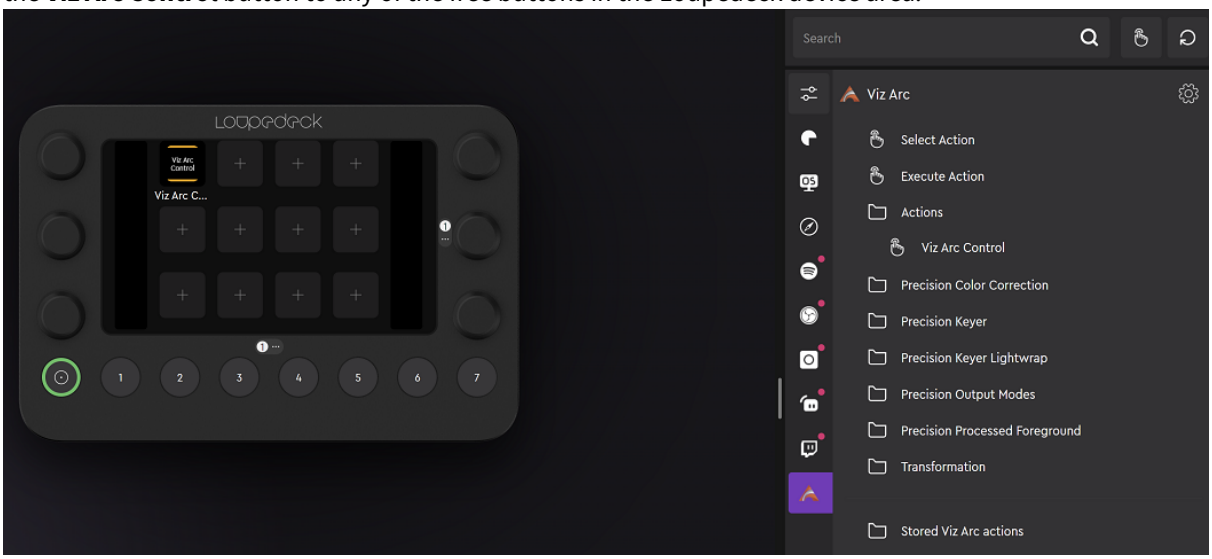


- Confirm the successful installation.

Copyright © 2025 Vizrt



- On the main page you can now find the Viz Arc plug-in. Select it and expand the **Actions** folder and drag the **Viz Arc Control** button to any of the free buttons in the Loupedeck device area.



Configuration

The configuration file *ArcSettings.json* can be found under *%localappdata%\Logi\LogiPluginService\PluginData\VizArc*.

```
{
```

Copyright © 2025 Vizrt

```
  "hostname": "localhost",
  "port" : 5004,
  "portREST" : 9004
}
```

The file contains the hostname and the port (for example, **localhost:5004**) of the running Viz **Arc In-App Web API** and the **Viz Arc REST Server**, the ports can be configured under the Communication section of the settings. Remove the port configurations or set them to 0, if the connection to either one of the services is not necessary or desired.

> **Note:** A restart of the Loupedeck software is required after every change to the configuration.

Logging

The log file *VizArc.log* of the plugin can be found under *%localappdata%\Logi\LogiPluginService\Logs\plugin_logs*.

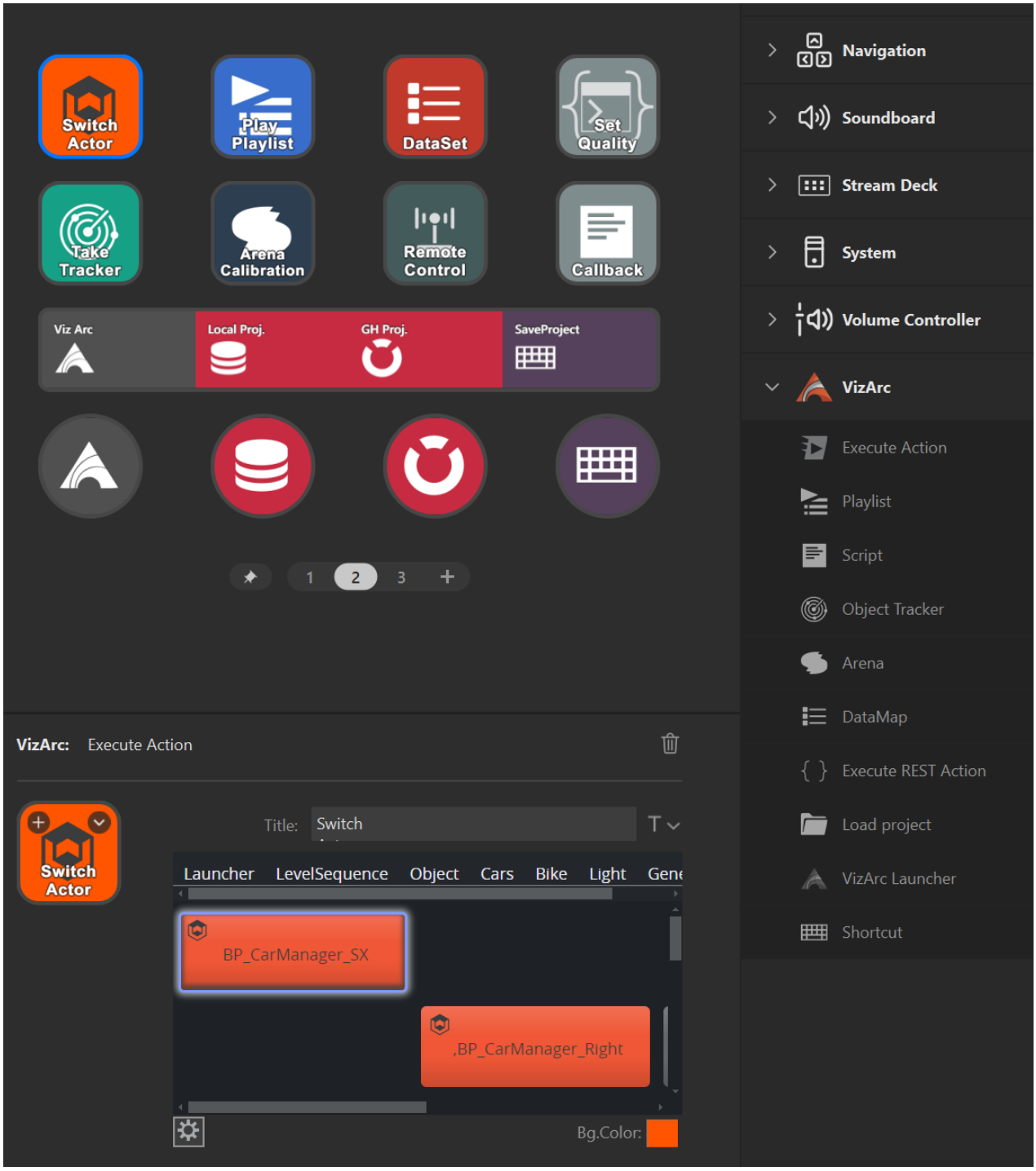
See Also

- Using the Loupedeck

Copyright © 2025 Vizrt

4.14 Stream Deck Integration

The Elgato Stream Deck plugin allows you to trigger various Viz Arc actions and shortcuts, such as Action Execution, Playlist control and much more.



- Installation
- Configuration

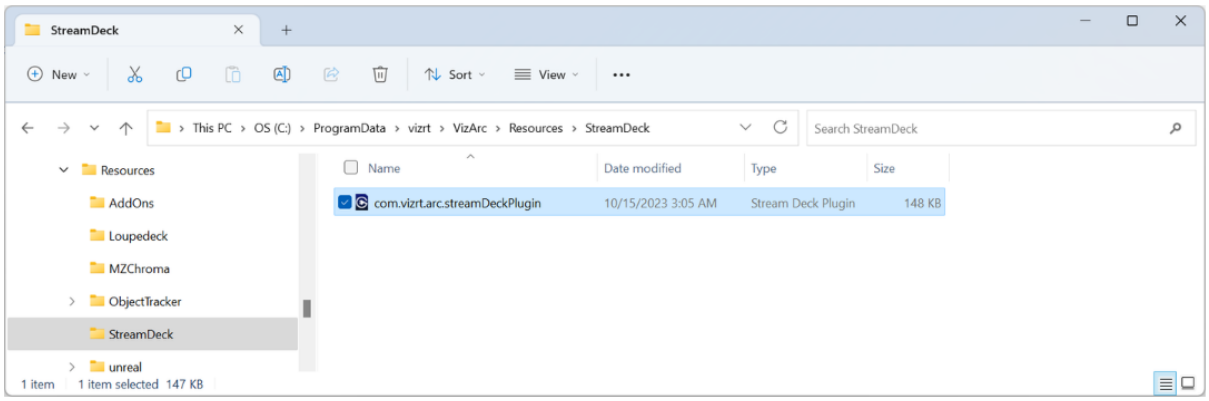
Copyright © 2025 Vizrt

- Internal vs. External Stream Deck Control
- Usage

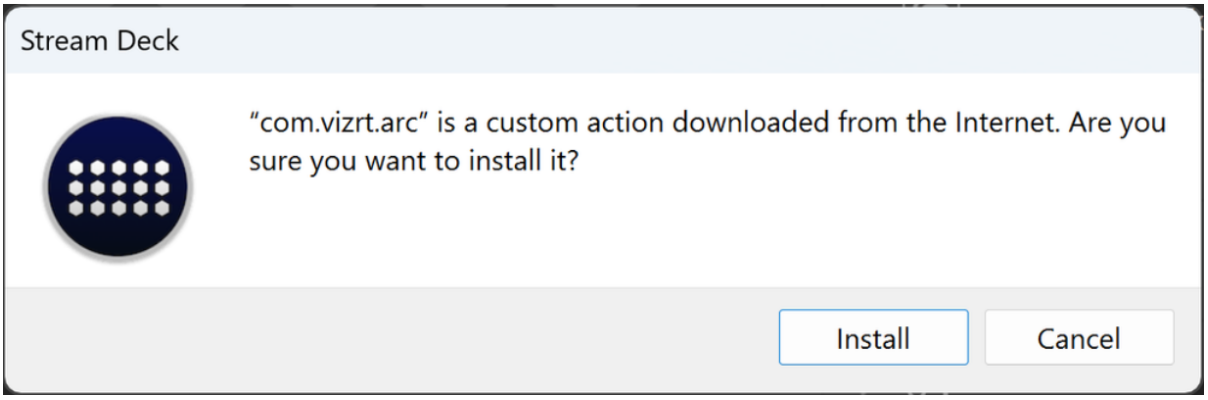
4.14.1 Installation

To use the Stream Deck plugin, download the Stream Deck for Windows software from here. The software does **not** necessarily need to be installed on the same machine as Viz Arc.

- Browse for the Viz Arc plugin contained in the file *com.vizrt.arc.streamDeckPlugin,* which can be found in *C:\ProgramData\vizrt\VizArc\Resources\StreamDeck* after the Viz Arc installation.

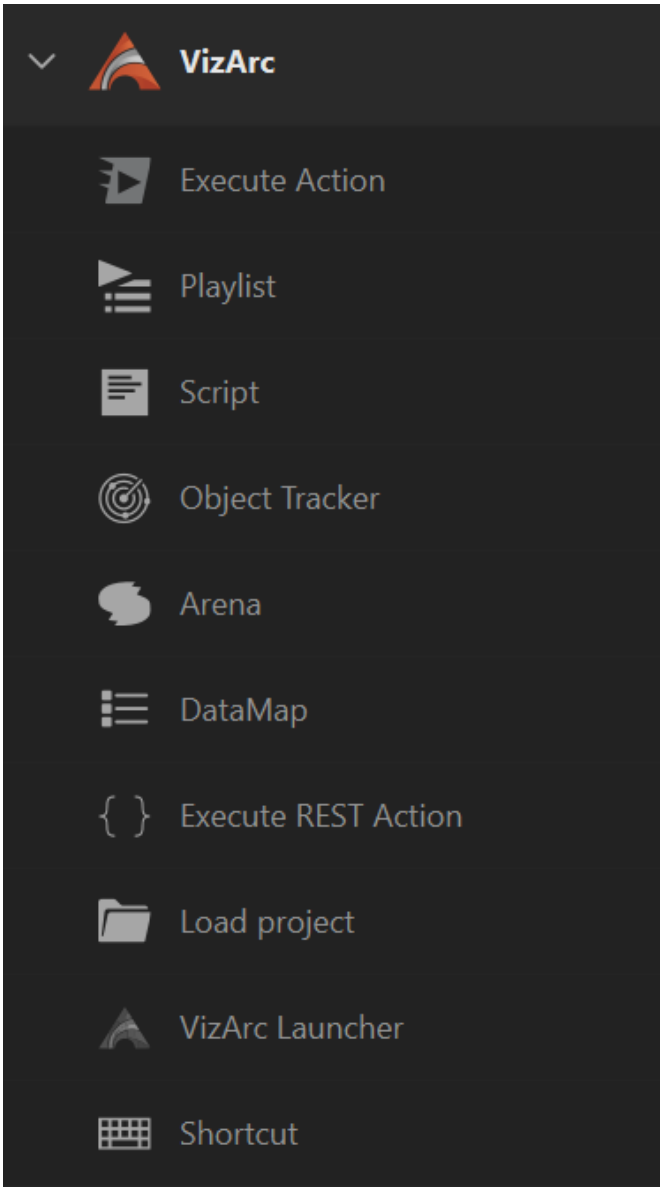


- Double click the file for installation. The Streamdeck application should show a prompt asking to confirm the installation.



- After the installation, you should see under the **Viz Arc** section the newly available actions.

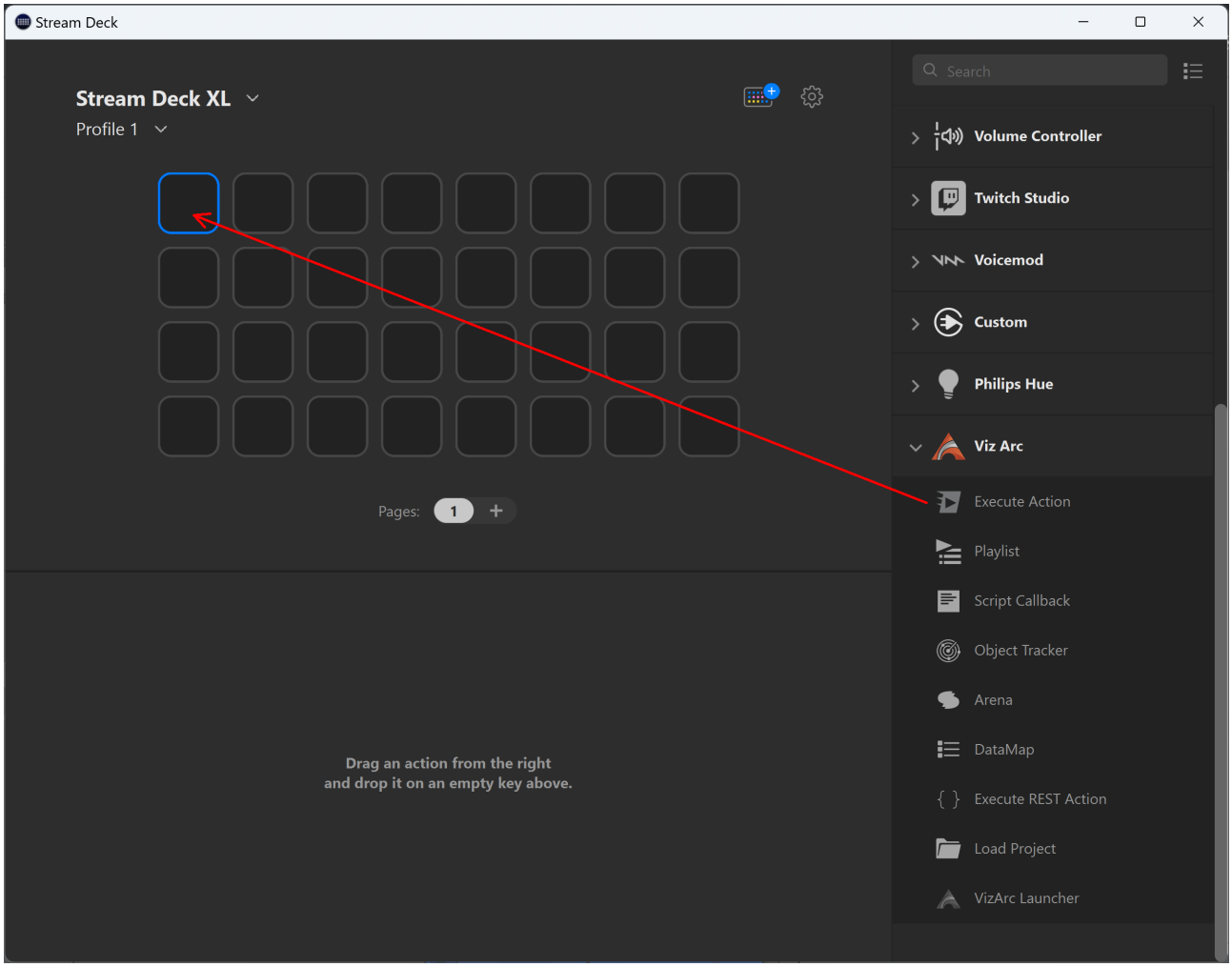
Copyright © 2025 Vizrt



4.14.2 Configuration

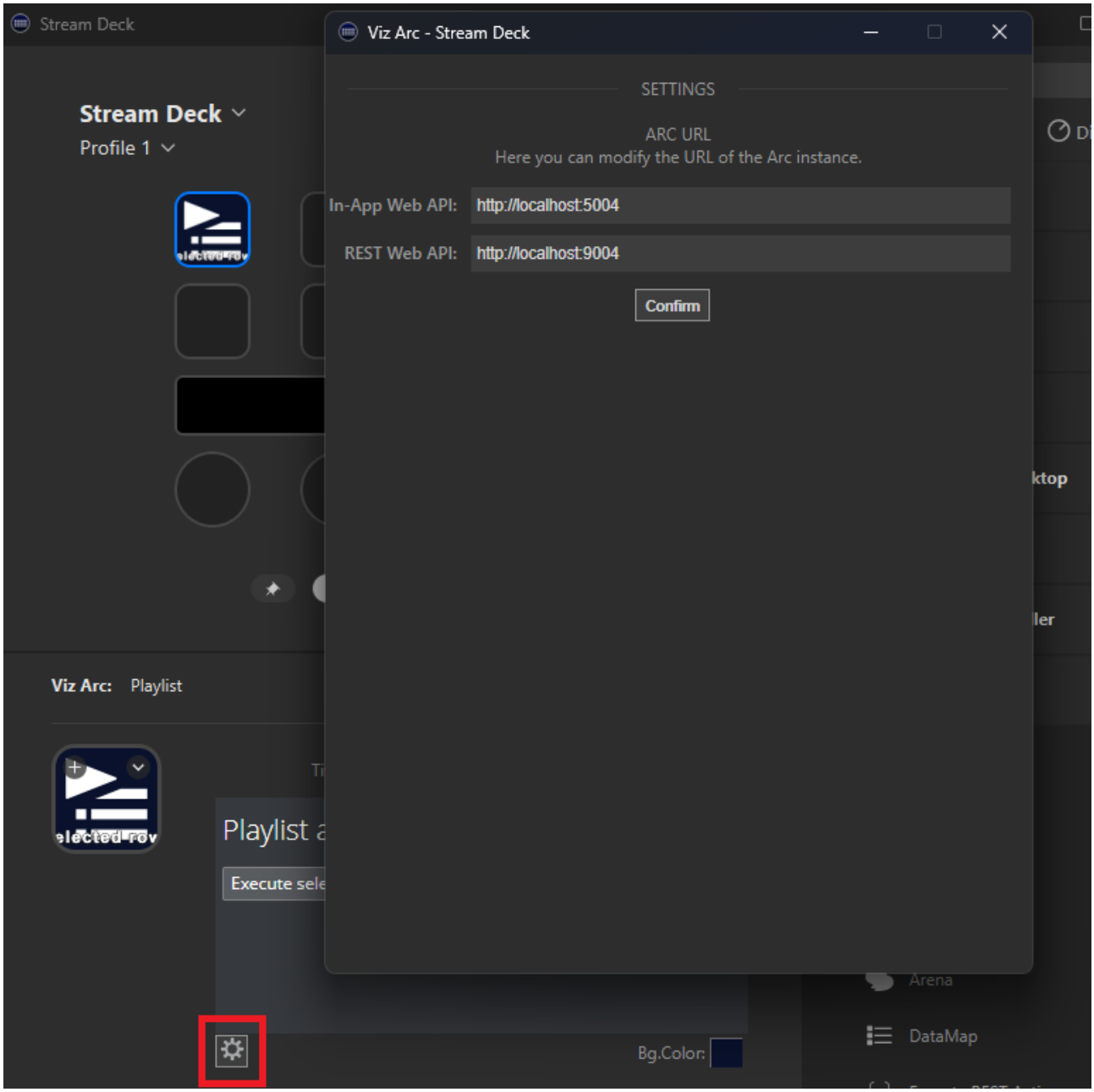
To configure the Viz Arc host(s), start to drag the **Execute Action** to any of the free Stream Deck keys.

Copyright © 2025 Vizrt



Click on the **Open Settings** button, if it does not open automatically to configure your Viz Arc **In-App Web API** and **REST Web API** endpoints.

Copyright © 2025 Vizrt



- **In-App Web API:** Defines the endpoint where the Viz Arc application is running (the default is http://localhost:5004).
- **REST Web API:** Defines the endpoint of the **Viz Arc REST Web API**. It is only required when using the **Execute REST Action**. The default is http://localhost:9004.

Info: These Viz Arc endpoints can be configured under the Communication settings.

Copyright © 2025 Vizrt

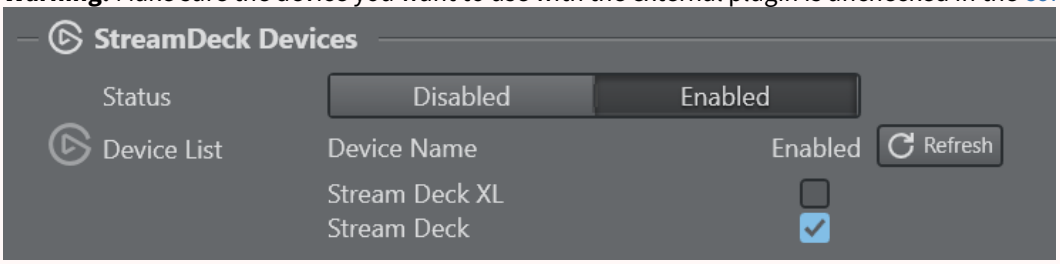
4.14.3 Internal vs. External Stream Deck Control

Please note that Viz Arc supports two ways on how to use a Stream Deck device.

1. Use the native integration, only available prior to version 1.9 which can be configured here.
2. Use the external integration using Elgato's Stream Deck desktop software, which is described in this section.

It is not recommended to use both integrations at the same time for a device, as there might be unexpected behavior.

Warning: Make sure the device you want to use with the external plugin is unchecked in the configuration.



4.14.4 Usage

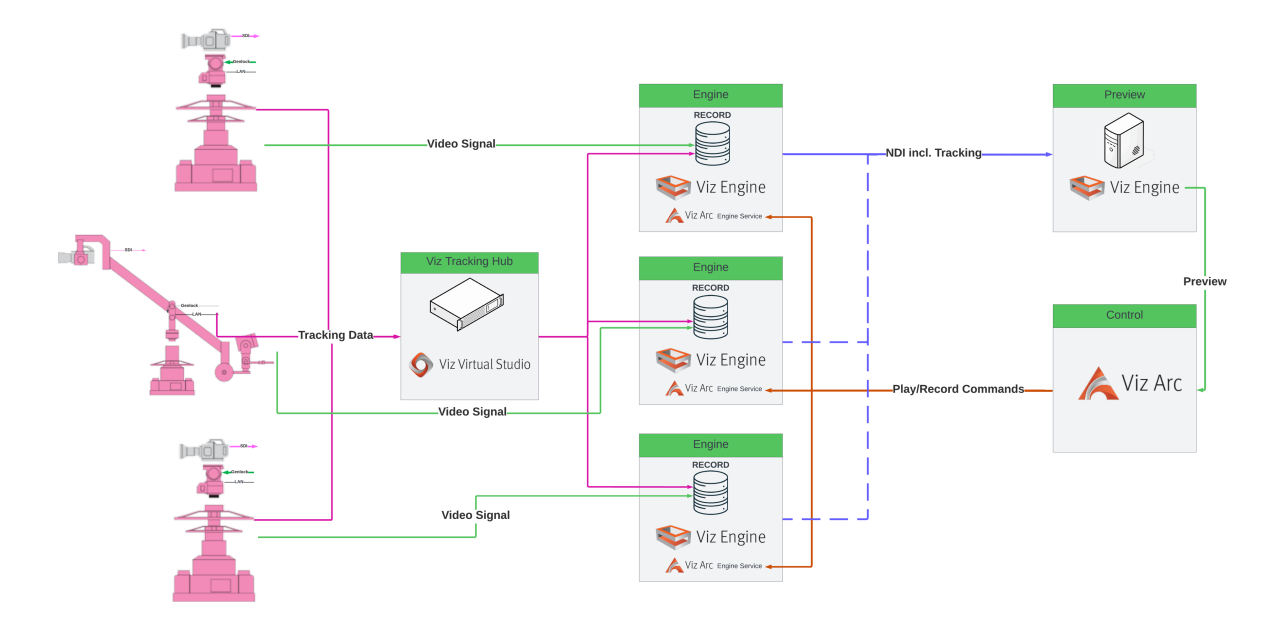
Read more about the usage here.

4.15 Preview Studio Configuration

4.15.1 Overview

The **Preview Studio** feature is a powerful component of the Viz Arc Engine Service that enables users to record Viz Engine's live input signal from one or more Viz Engines, and camera tracking data from Viz Tracking Hub. The recordings can be replayed later on.

This functionality creates a virtual sandbox environment where operators can test, refine, and preview their graphics without requiring live video input or tracking hardware.



- Overview
  - System Requirements
- Configuration
  - Viz Engine Configuration
  - Preview Viz Engine Configuration
  - Viz Arc Configuration

System Requirements

Preview Studio integrates with the existing Viz Arc ecosystem, requiring:

- One or more Viz Engine to record the input signal from. Viz Arc Engine Service must be installed beside each Viz Engine instance.
- A Virtual Studio rig instance that sends data to the Viz Arc Engine Service (default port 3001).
- Sufficient storage for video and tracking data. The video and tracking data that are being recorded, are stored according to the location specified in the *appSettings.json* file, found in the Viz Arc Engine Service installation folder.

Copyright © 2025 Vizrt

- A Viz Engine instance that acts as Preview Server, on which the recordings are displayed. This is ideally a Viz Engine configured as NDI engine on the local machine where Viz Arc is running.

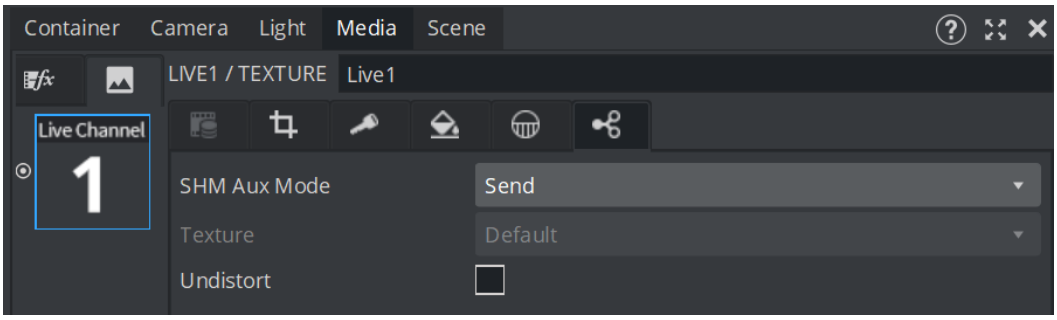
4.15.2 Configuration

Viz Engine Configuration

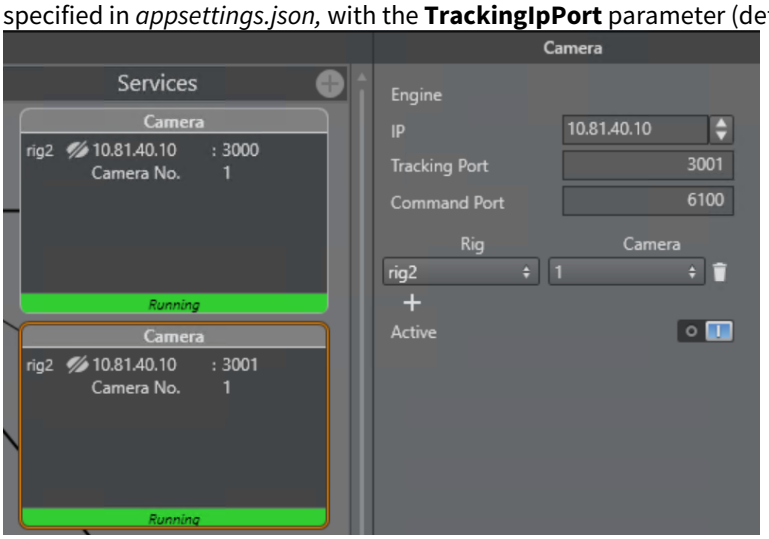
1. **Configure SMURF Input:** Configure Viz Engine SMURF input option. The key can be found and changed in the Viz Engine configuration file, by default, the key for live input1 is *viz_01_live1_aux*:

```
#* LiveIn1.SHMAuxKey: Default=viz_01_live1_aux
# LiveIn1.SHMAuxKey = viz_01_live1_aux
```

2. **Enable SMURF Send in Scene:** Sets the **SHM Aux Mode** to **send** in the scene.



3. **Configure Studio Manager:** Sets up a rig dedicated to the Viz Arc Engine Service. Direct it to the port specified in *appsettings.json,* with the **TrackingIpPort** parameter (default: 3001).



Preview Viz Engine Configuration

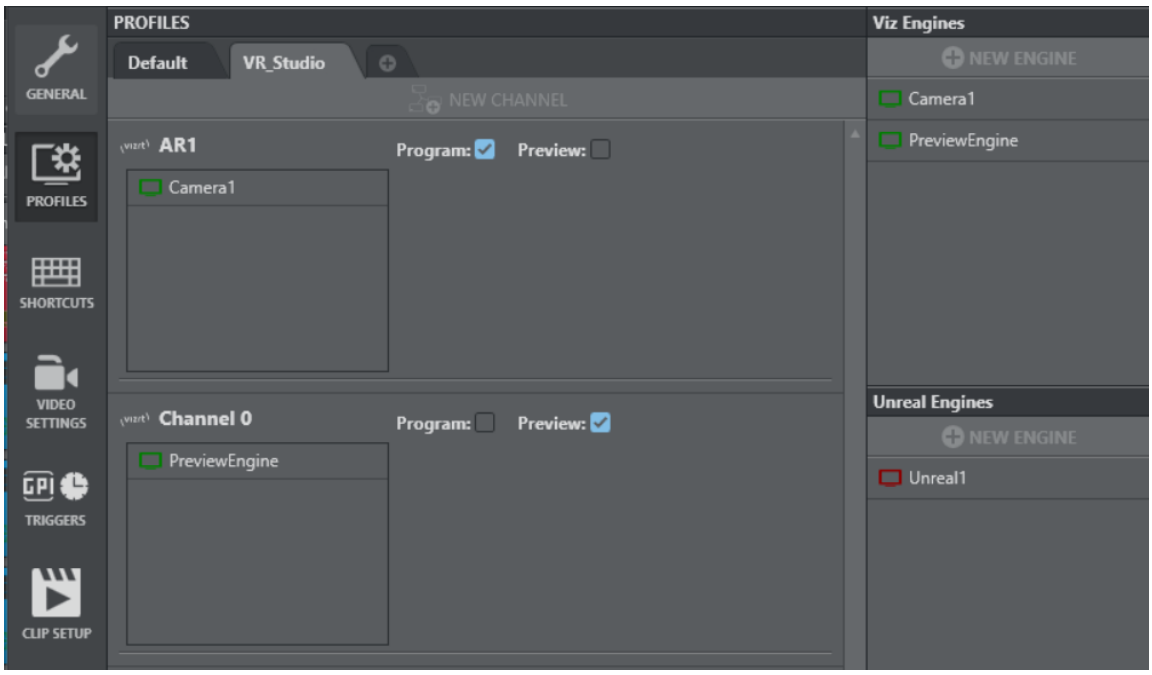
It is recommended to use a dedicated Viz Engine used as Preview Engine, that receives the NDI playback streams.

- **Enable SMURF/NDI:** Configure Viz Engine to use the tracking data coming from the NDI stream instead of Tracking Hub. The config file default location is under *C:\ProgramData\vizrt\VizEngine\VizEngine-0.cfg,* and the parameter set to *use_trackinghub = 3*.

Copyright © 2025 Vizrt

Viz Arc Configuration

Define a Profile: In Config → Profiles, create a profile that includes all engines being recorded from, and the Viz Engine that functions as Preview Server, by checking the **Preview** box.



Please note that the ***Engine Service Port*** parameter must be properly configured in order for Viz Arc to communicate with the remote Engine Services.



When the set up is complete, the usage instructions can be found on Preview Studio.

Copyright © 2025 Vizrt

5 Working with Viz Arc

This section covers the following topics:

- General Workflow
- User Interface
- 4 Point Calibration
- Supported Action Types
- Builder Mode
- APIs
- Data Integration
- Integrations
- Engine Status Widget
- Graphic Hub Browser
- Import and Export
- Multizone Chroma Keying
- OCR
- On Air Mode
- Playlist
- Preview Studio
- Preview
- Precision Keyer
- Program and Preview
- Projects
- Templates

Copyright © 2025 Vizrt

5.1 General Workflow



This section provides an overview of the general workflow in Viz Arc.

Viz Arc is based on Projects. A project defines the scene or scenes to be used in any given production as well as the different buttons called Actions that control the desired scene properties, such as transformations or triggering animations.

The interface has two main modes, Builder and On Air:

- **Builder mode:** Users are able to configure and build all the needed actions for the operation.
- **On Air mode:** Users can execute and manage existing actions, but have limited functionality on creating new ones.

Creating and working with projects is simple, just follow the steps below.

- Add a scene to a project to start creating actions that control your scene.
- Actions can be created by dragging and dropping the property you wish to control such as Transformation, Text, Alpha, etc. from the Control Channels, Tree or Directors sections to the Actions Panel. For more details, see create actions and supported action types.
- Optionally Configure the different tracked cameras, zero point and transformation properties of the different elements within your Virtual Studio or Augmented Reality elements by importing an AutoCAD (*.dxf*). For outdoor productions, configure directly over a map in the Set section of the interface.
- Go On Air and initialize your project to load all the imported scenes into the selected Channels and start executing your actions.

Copyright © 2025 Vizrt

Note: Viz Arc also lets you write your own custom Scripts.

Copyright © 2025 Vizrt

5.2 User Interface

The Viz Arc user interface is divided into the following main sections:

- Builder Mode
  - Actions
  - Set
  - Script
- On Air Mode
- Top Tool Bar
- Status Bar

5.2.1 Builder Mode



The Operations Toolbar is at the top right corner When working in Builder mode, you can configure and build all the actions required for On Air operation.

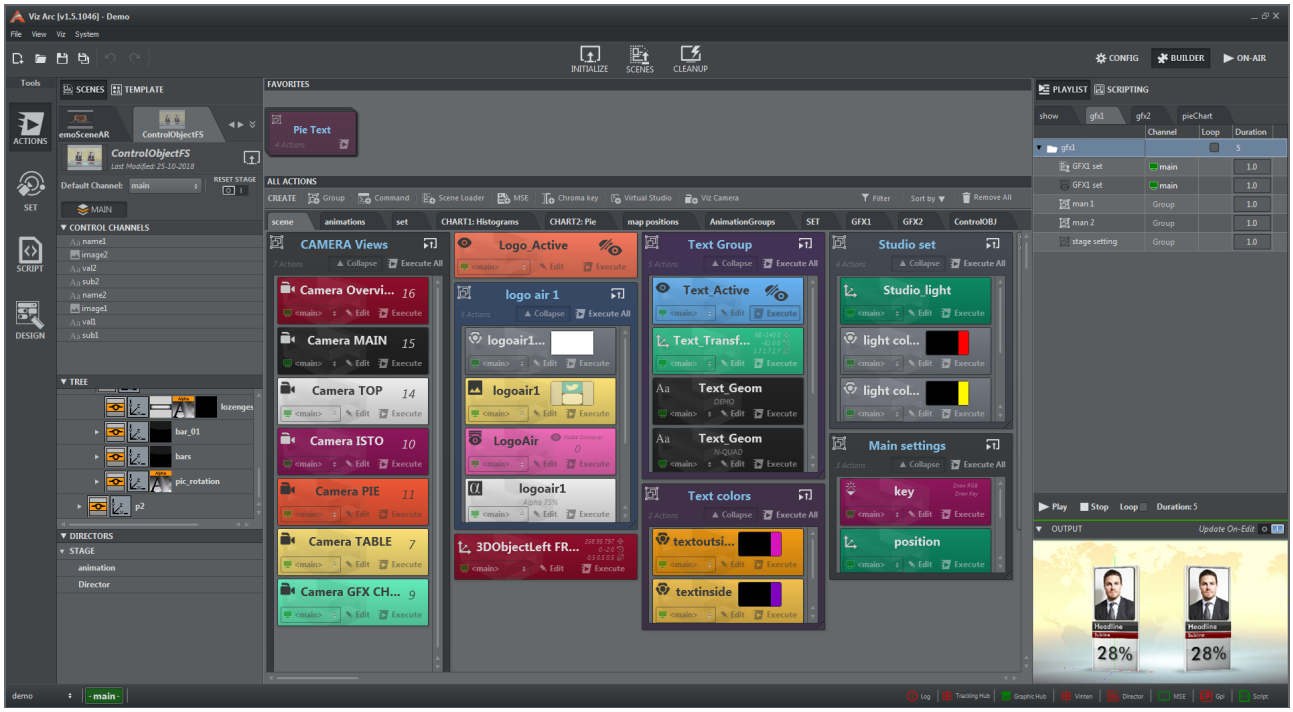


In builder mode, three main views contain tools for creating Projects:

- Actions
- Set
- Script

Copyright © 2025 Vizrt

Actions

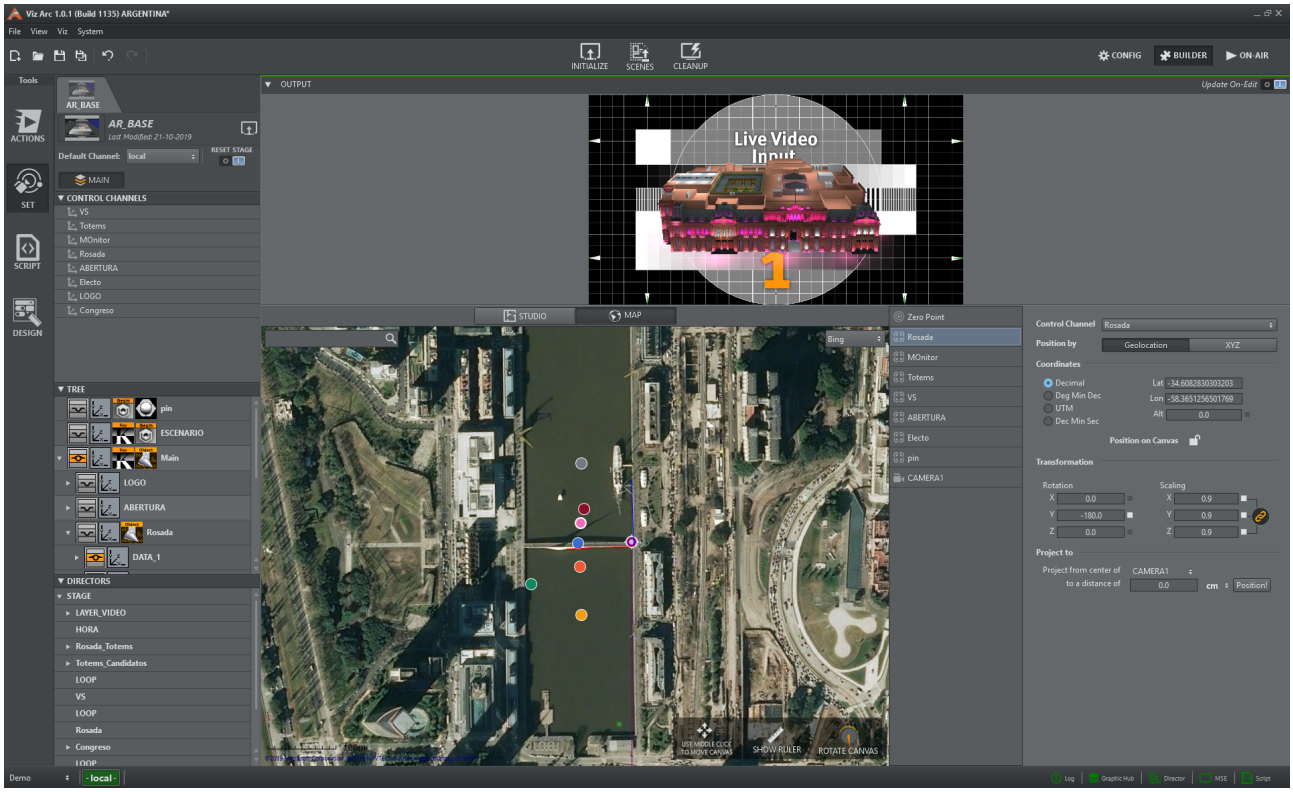


This is the part of the UI where most of the work is done when creating Projects.
The panel contains all the tools required to:

- Import one or more scenes to be controlled by Viz Arc.
- Create actions to control properties such as position, rotation, scaling, modifying text, trigger animations, etc. of the different scene elements.

Copyright © 2025 Vizrt

Set



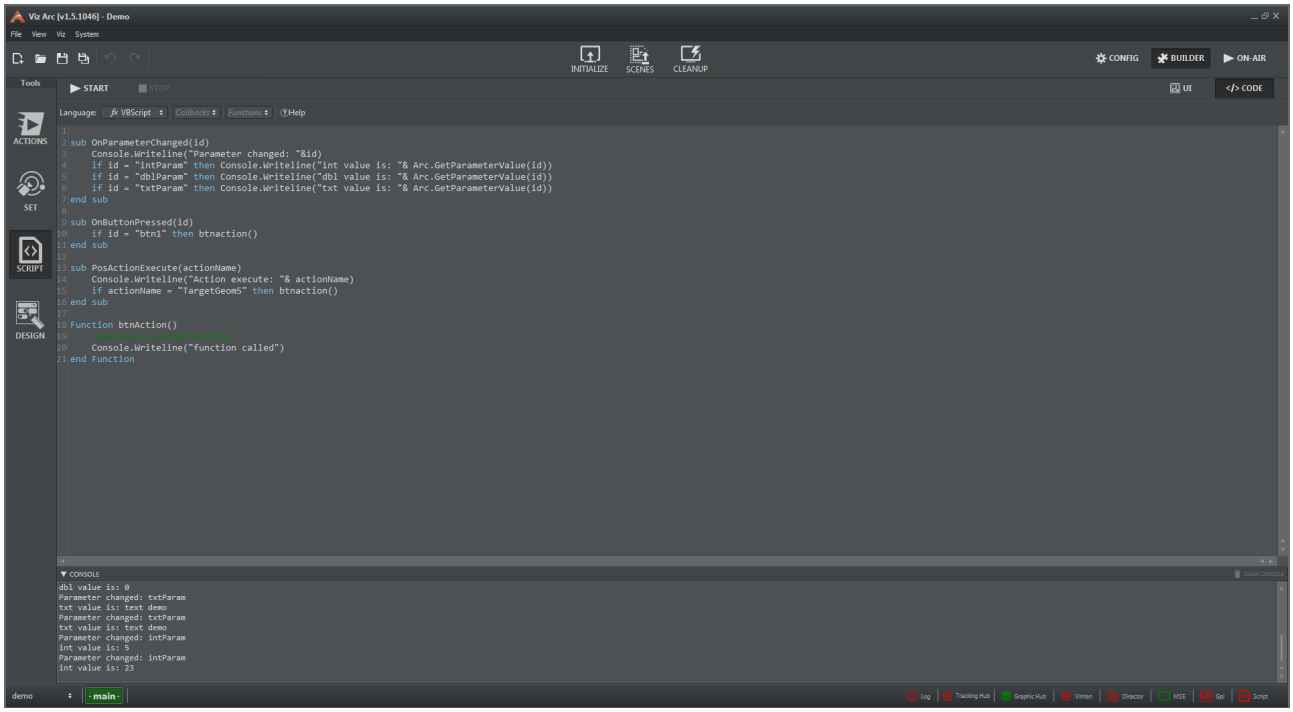
One of the key features of Viz Arc is the ability to configure all your virtual studio and augmented reality elements' positions in relation to your physical environment and tracked cameras. For indoor productions, you can import an AutoCAD project (*.dxf*) to set up the different elements or work over a map when outdoors.

The panel contains all the tools required to:

- Position, Scale and Rotate elements to fit your physical environment.
- Set the position of your zero point.
- Setup tracked cameras.
- Create chroma key polygons.

Copyright © 2025 Vizrt

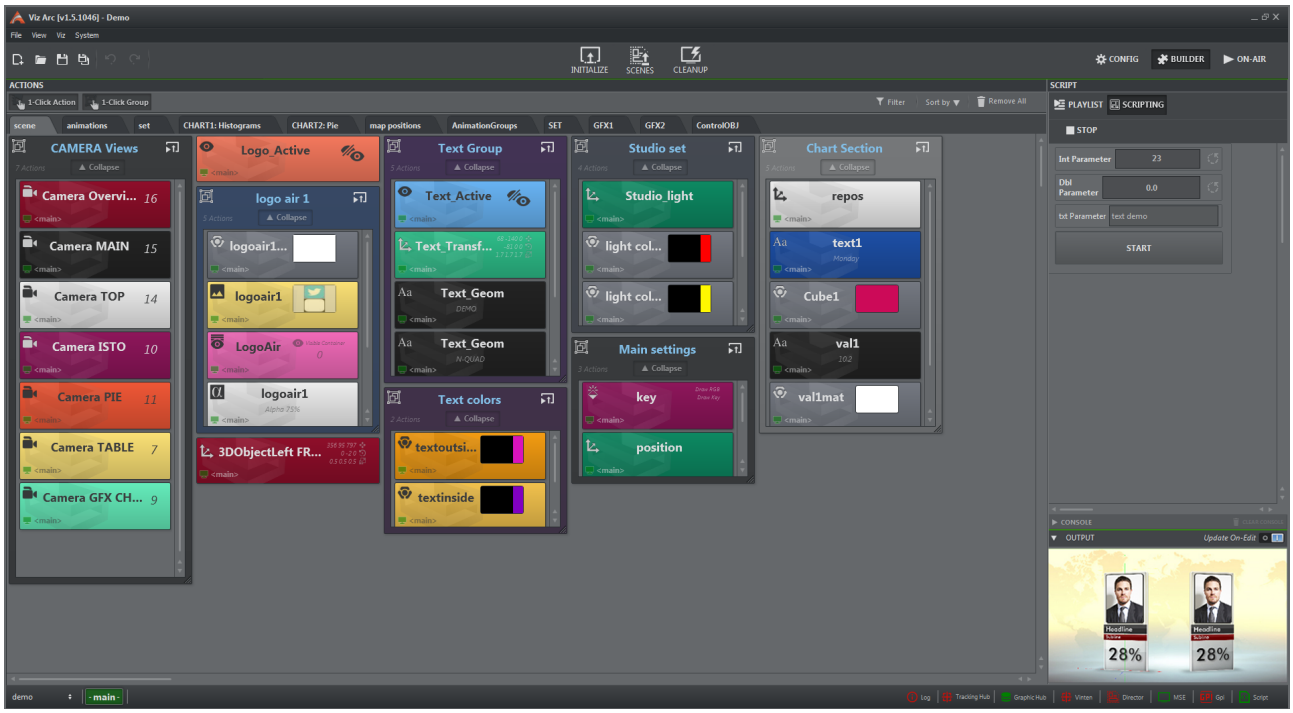
Script



Viz Arc includes a built-in scripting engine for connecting to a database or granting additional control to operators. Custom scripts can be written from this section of the interface. Scripts can include form elements such as text boxes, buttons, etc. that are available to users from the Actions panel or when working in On Air mode.

Copyright © 2025 Vizrt

5.2.2 On Air Mode



This part of the interface is where operators work with previously created Viz Arc Projects. On Air mode provides a simple UI where operators can focus on controlling the live show to execute and manage existing actions. When working in On Air mode, users have limited functionality for creating new actions.

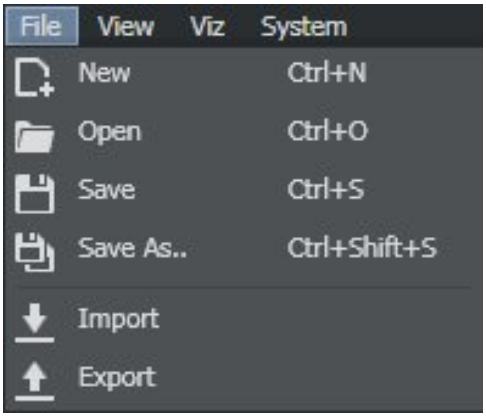
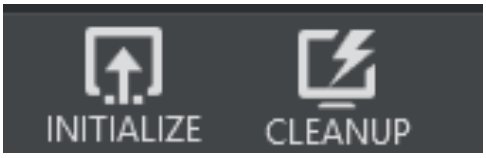
> **Note:** On Air users can edit, execute, delete and duplicate existing actions or add general project actions such as groups, etc.

5.2.3 Top Tool Bar



Located at the top of the main window, the File Menu is composed of:

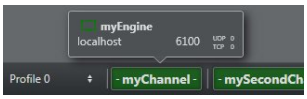
Copyright © 2025 Vizrt

	Main Menu and Project Toolbar (**File**, **View**, **Viz**, **System**)
	Project Tools (**New**, **Open**, **Save**, ..)
	Scenes Toolbar section (**Initialize**, **Cleanup**),
	Operations Toolbar section (**Config**, **Builder**, **On-Air**).

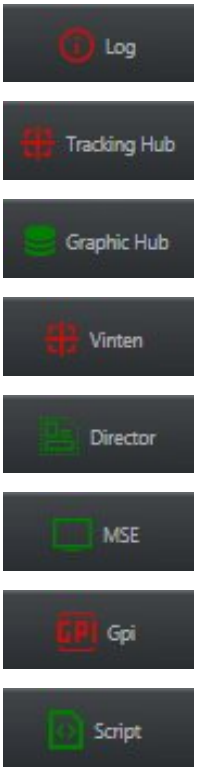
5.2.4 Status Bar



Located at the bottom of the main window, the status section is composed of:

	**Profiles and Channels section** (left) • Connection Status for Channels of the selected Profile

Copyright © 2025 Vizrt

	**Connections section** (right)
	• Log list
	• Connection status for
	  • Tracking Hub
	  • Graphic Hub
	  • Vinten
	  • Director
	  • MSE
	  • GPI Board
	• Compilation Status of Project Script

See Also

- Supported Action Types

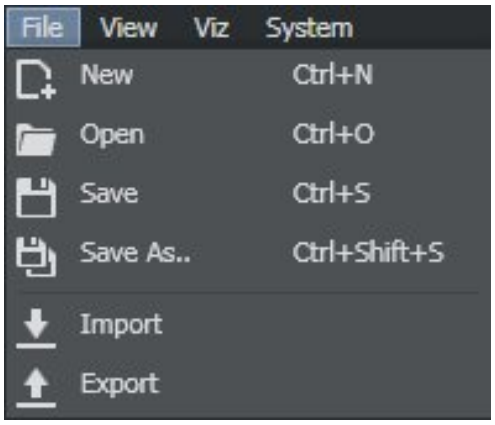
5.2.5 Main Menu and Project Toolbar

The Main menu contains the following sections:

- File
- Projects
- View
- Viz
- System

File

All Project creation and management related actions are found in this menu.



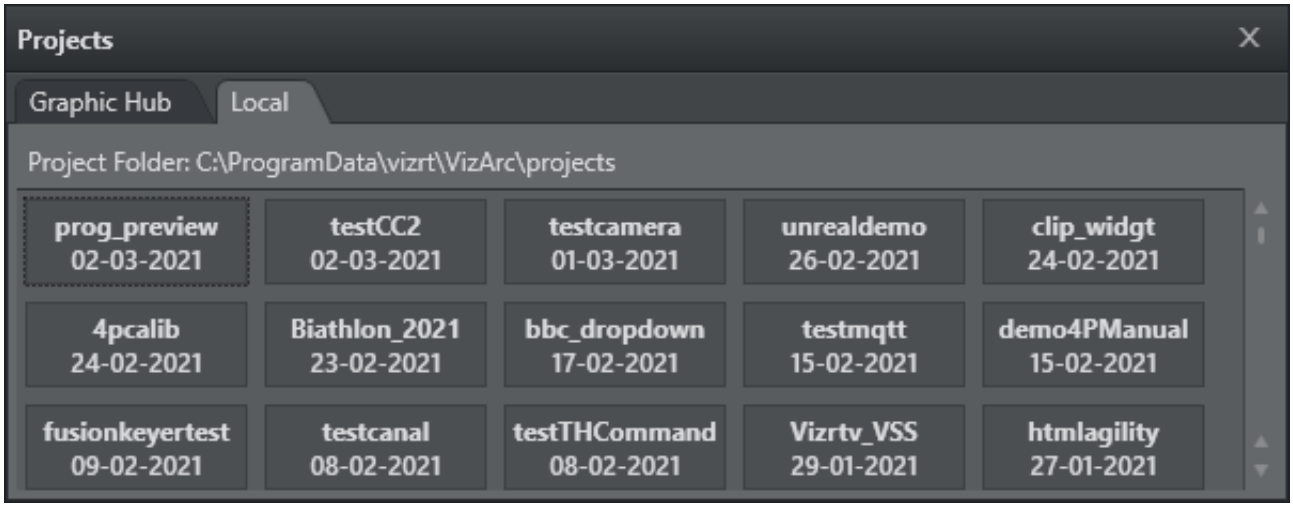
- **New (CTRL + N):** Creates a new blank project.
- **Open (CTRL + O):** Opens existing project.
- **Save (CTRL + S):** Saves the current project.
- **Save As (CTRL + SHIFT + S):** Saves project with a different name.
- Import**:** Imports a Viz Arc archive (*.arc* file).
- Export**:** Exports a Viz Arc archive (*.arc* file).

Note: The same operations are also available from the Project toolbar:

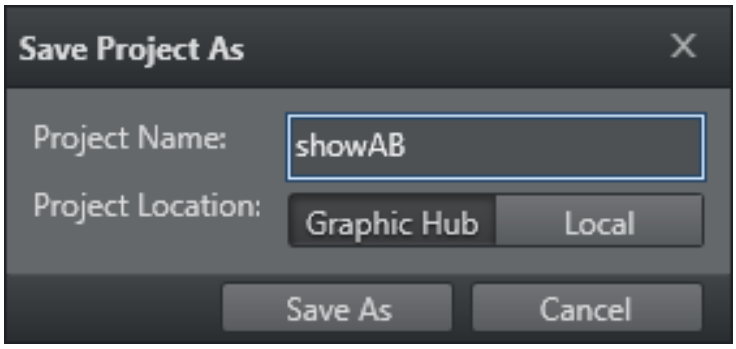
Projects

Projects can be located on the local file system or on Graphic Hub. When opening a project you can choose where to load it from, selecting from either the **Graphic Hub** tab and the **Local** tab.

Copyright © 2025 Vizrt

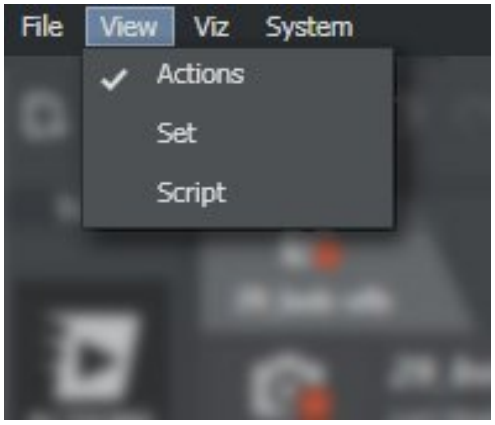


When saving a project, you can choose whether to store it locally or on Graphic Hub using the respective radio button.



View

This menu lets you switch between the three main views when working in Builder mode.



- **Actions**: Switches to the Actions view in the main window.
- **Set:** Switches to the Set view in the main window.
- **Script:** Switches to the Script view in the main window.

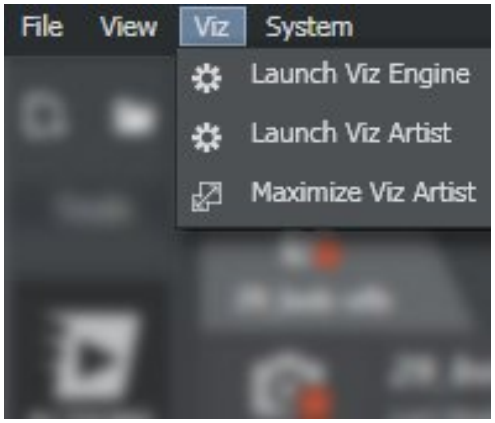
Copyright © 2025 Vizrt

You can also switch between views using this toolbar:



Viz

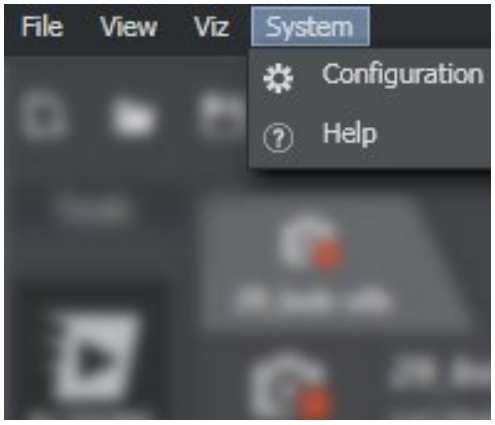
This menu contains actions relating to Viz systems.



- **Launch Viz Engine:** Launches Viz Engine.
- **Launch Viz Artist:** Launches Viz Artist (if licensed).
- **Maximize Viz Artist:** Switches to the Viz Artist window.

Copyright © 2025 Vizrt

System

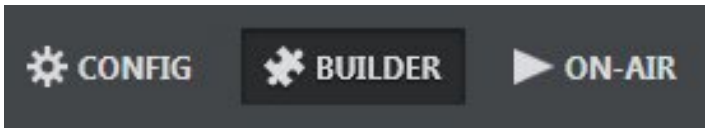


- **Configuration:** Opens the configuration section in a new window.
- **Help:** Opens the present manual.

Note: The configuration panel can also be opened by clicking CONFIG at the top right.

5.2.6 Operations Toolbar

This page explains the functionality of the different actions available in the **Operations Toolbar.**

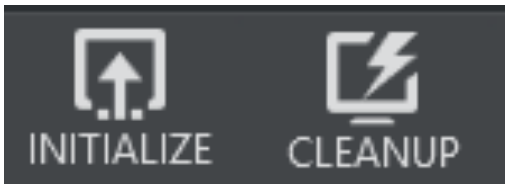


- **CONFIGURATION:** Opens the configuration window.
- **BUILDER:** Switches to the project editing mode.
- **ON-AIR:** Switches to the Live section, where some actions are limited to the live show.

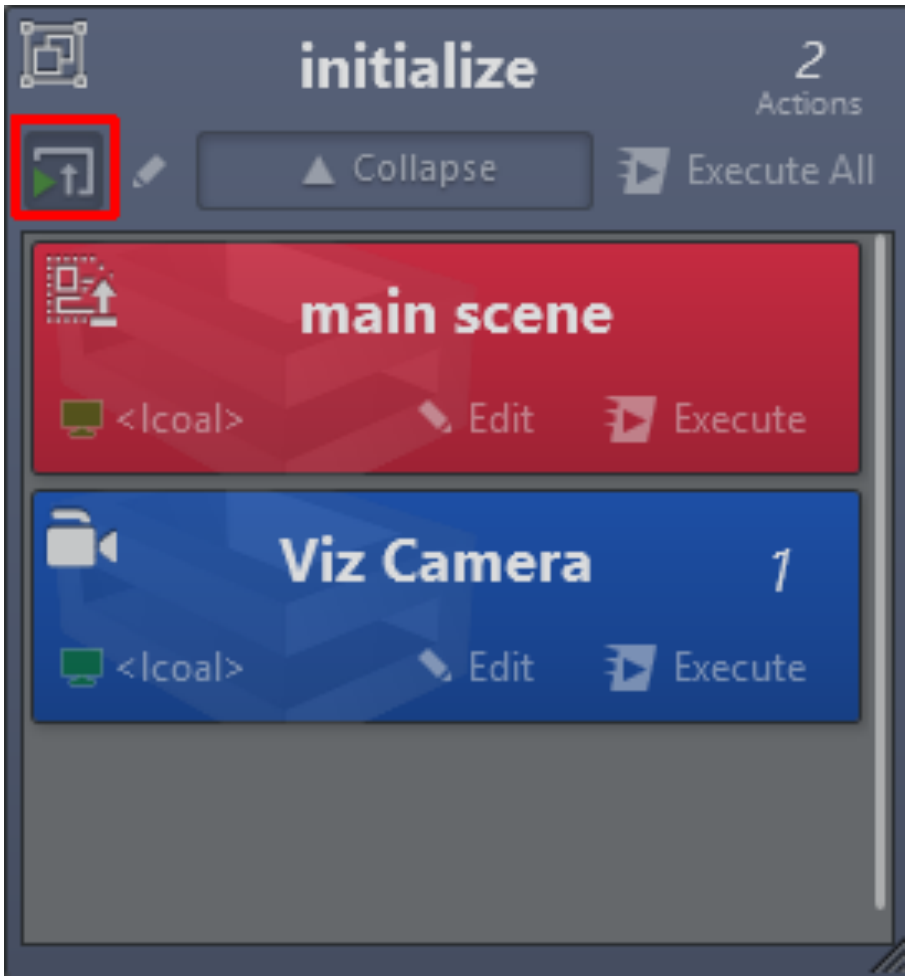
Copyright © 2025 Vizrt

5.2.7 Scenes Toolbar

This section describes actions in the **Scenes Toolbar.**



- **INITIALIZE:** Loads all scenes included in the open project to the configured channels and executes all groups tagged with the *Execute on Initialize* toggle.
- **CLEANUP:** Unloads any loaded scenes, images, etc. from all engines defined in all channels from the selected profile.

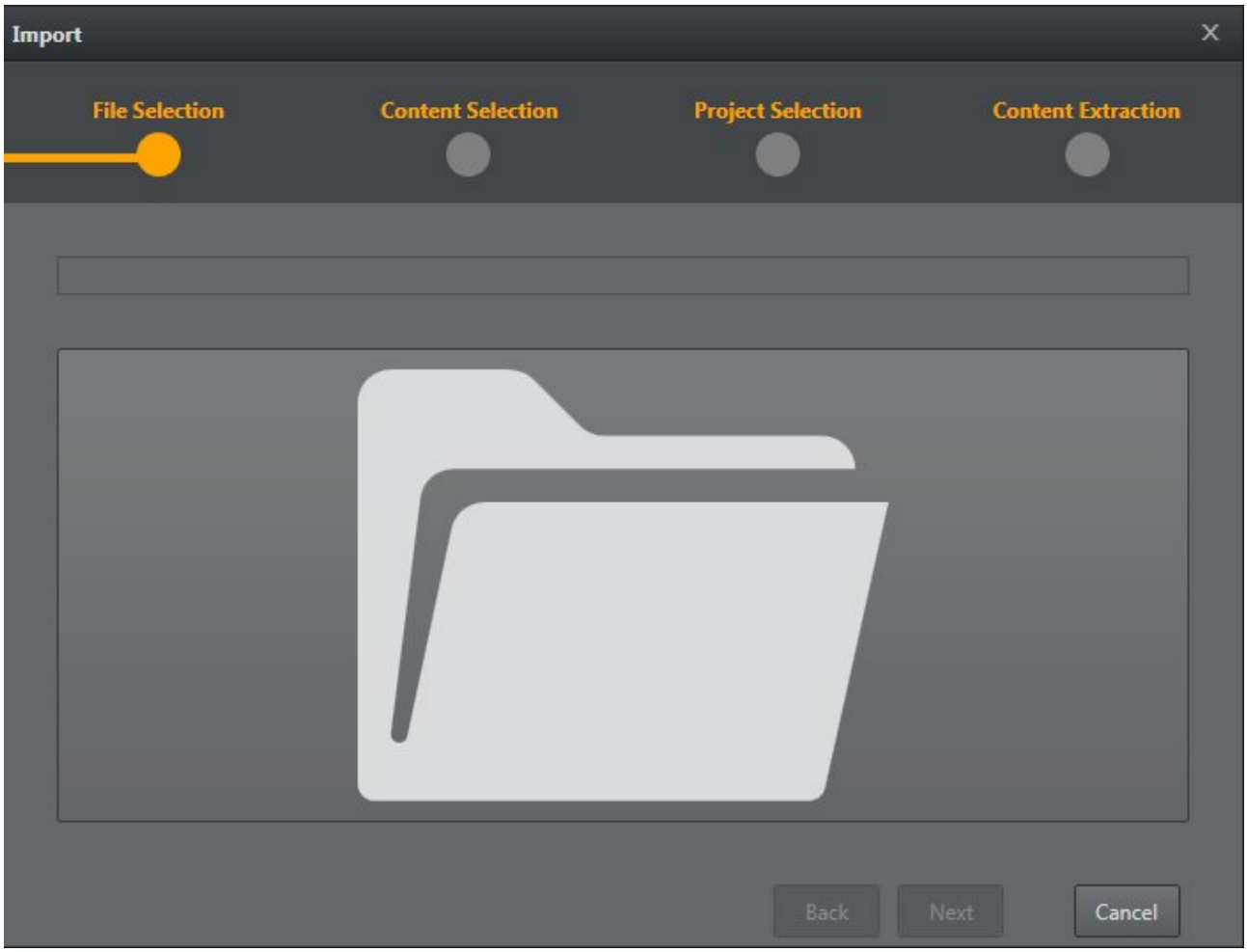


The screenshot above shows a group that is executed with the **initialize** button, since its *Execute on Initialize* toggle is active.

Copyright © 2025 Vizrt

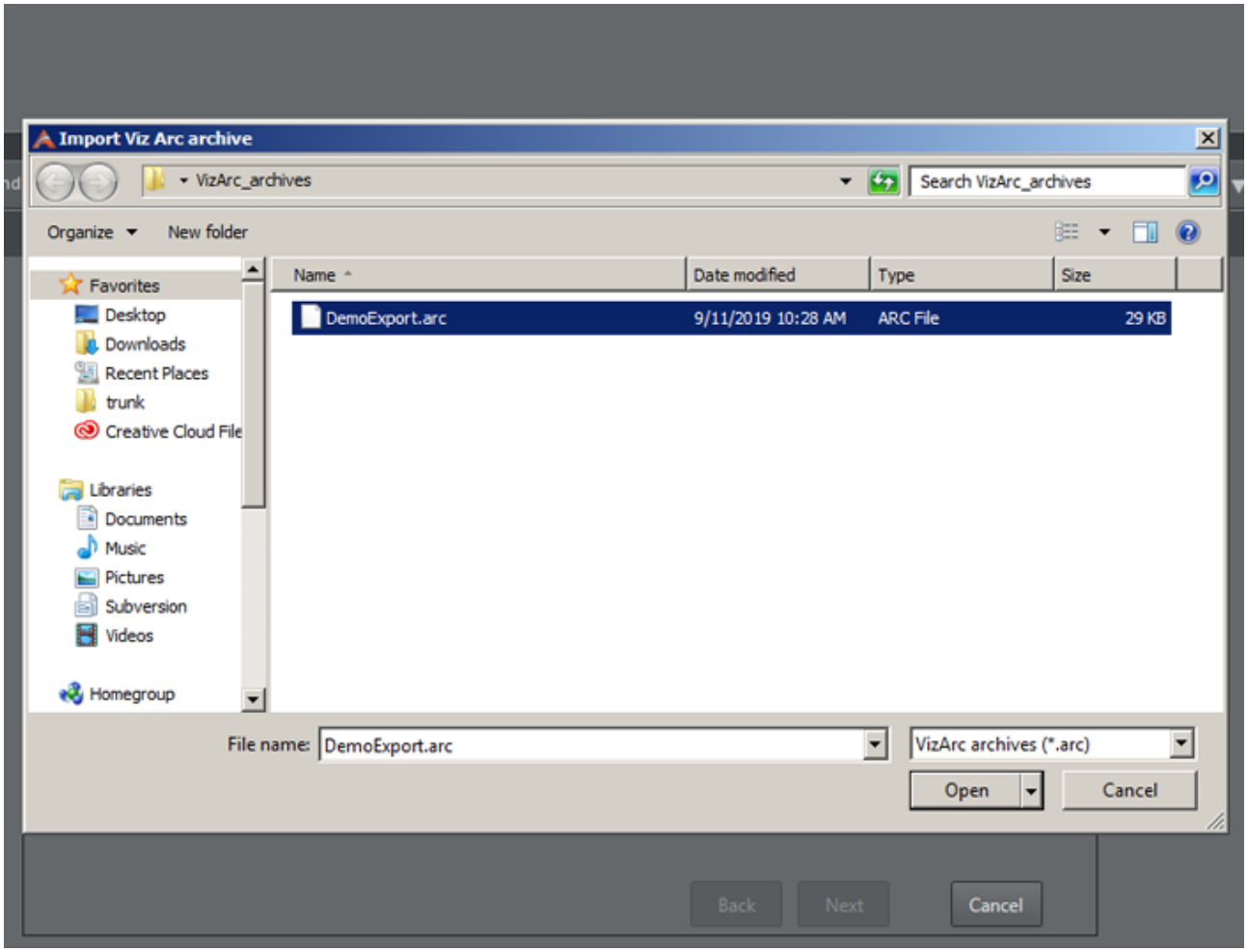
5.2.8 Import

Select **File > Import** to make the window below appear:



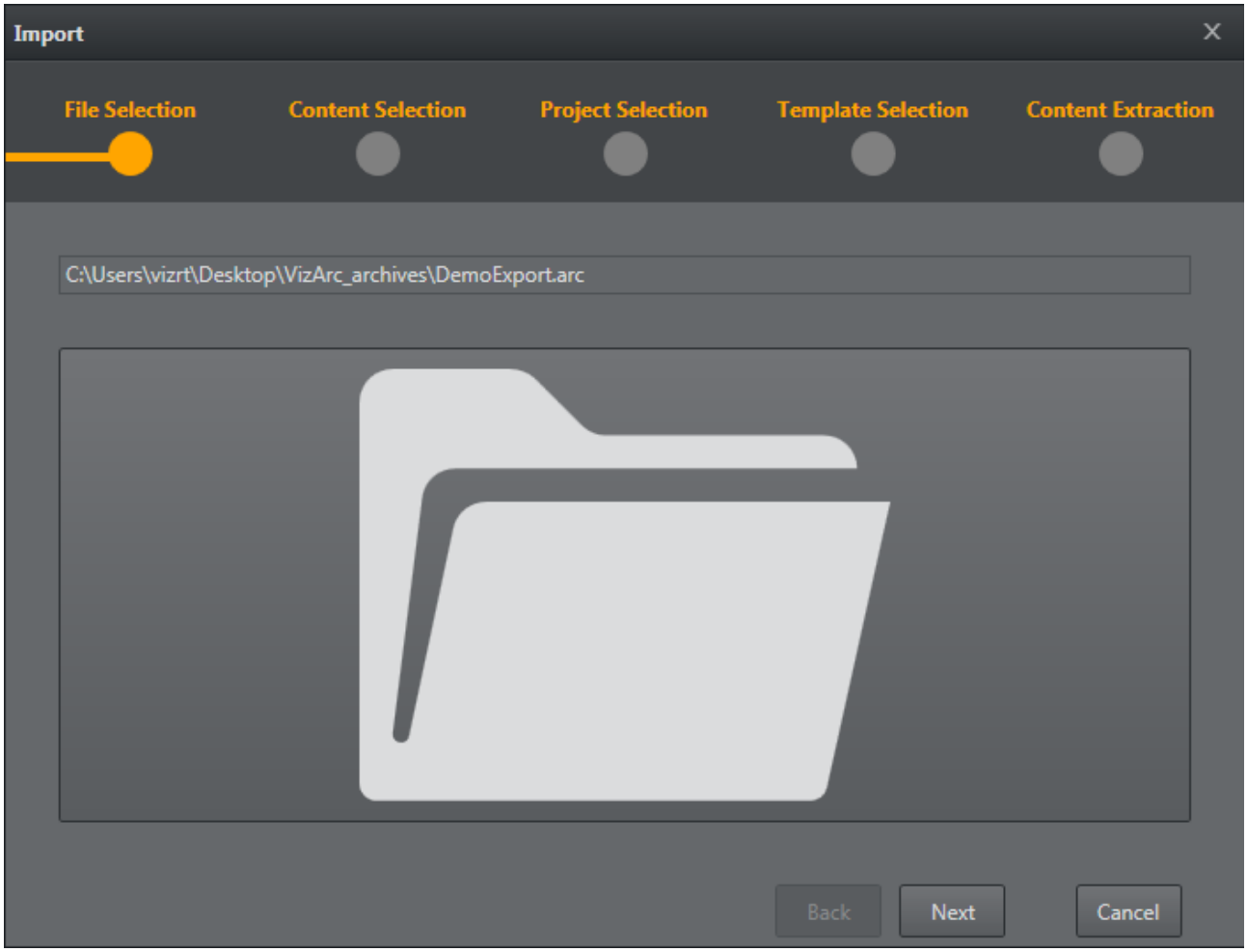
To open the selection dialog window, press the big folder icon.

Copyright © 2025 Vizrt



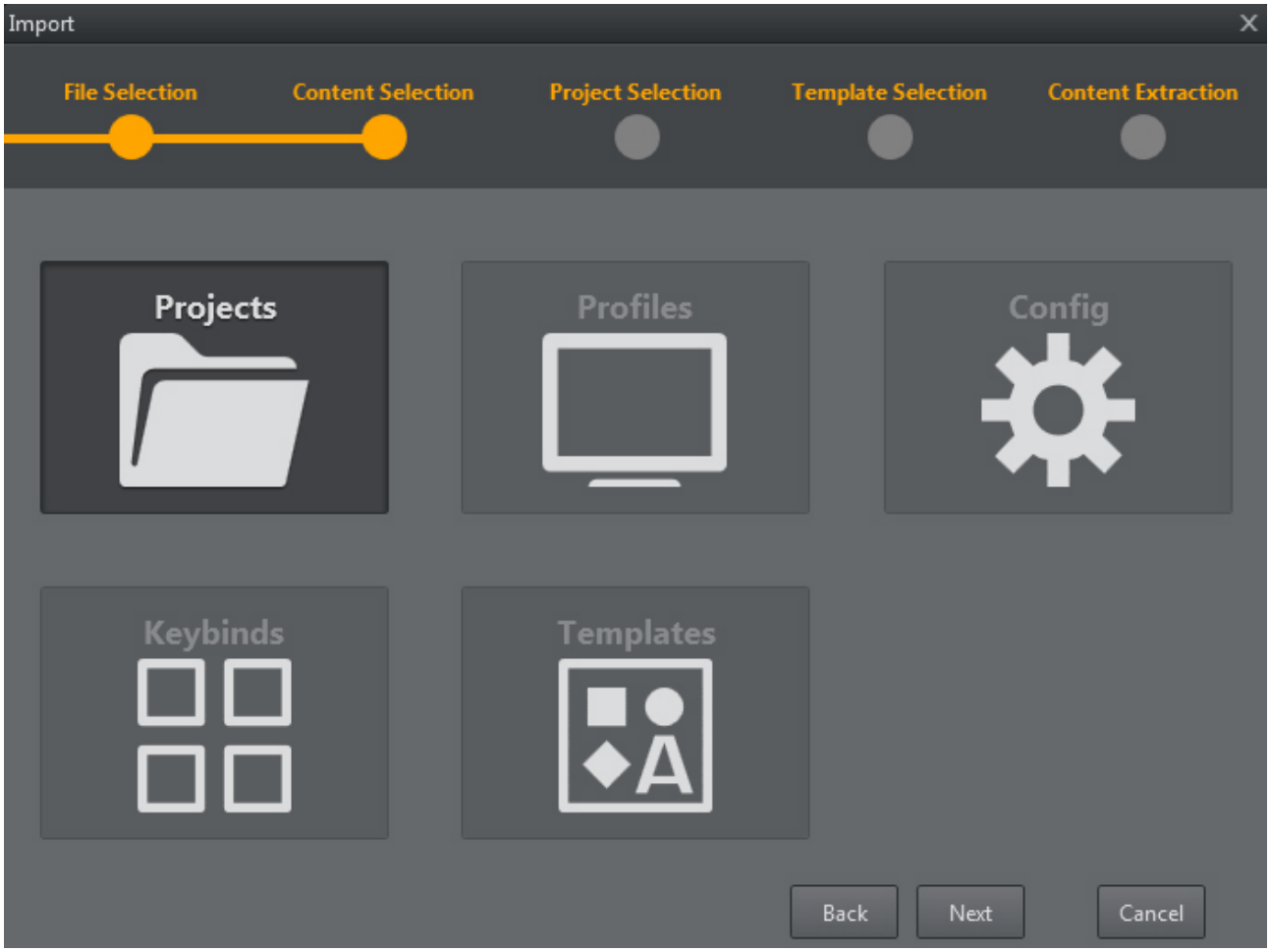
After selecting the file, you can proceed with the import steps.

Copyright © 2025 Vizrt



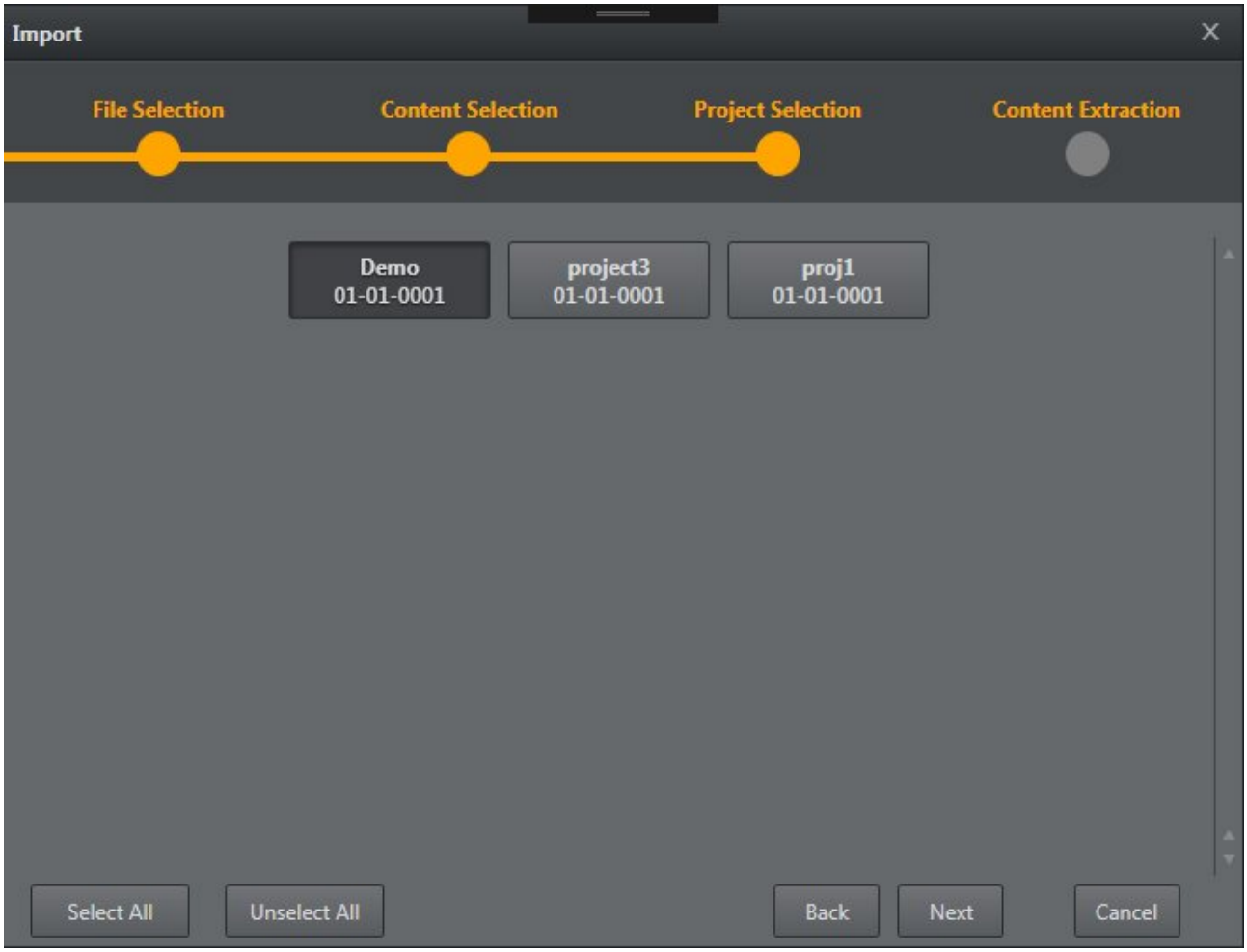
Select the elements to import (**Projects**, **Profiles**, **Config** or **Keybinds**) and click **Next**.

Copyright © 2025 Vizrt



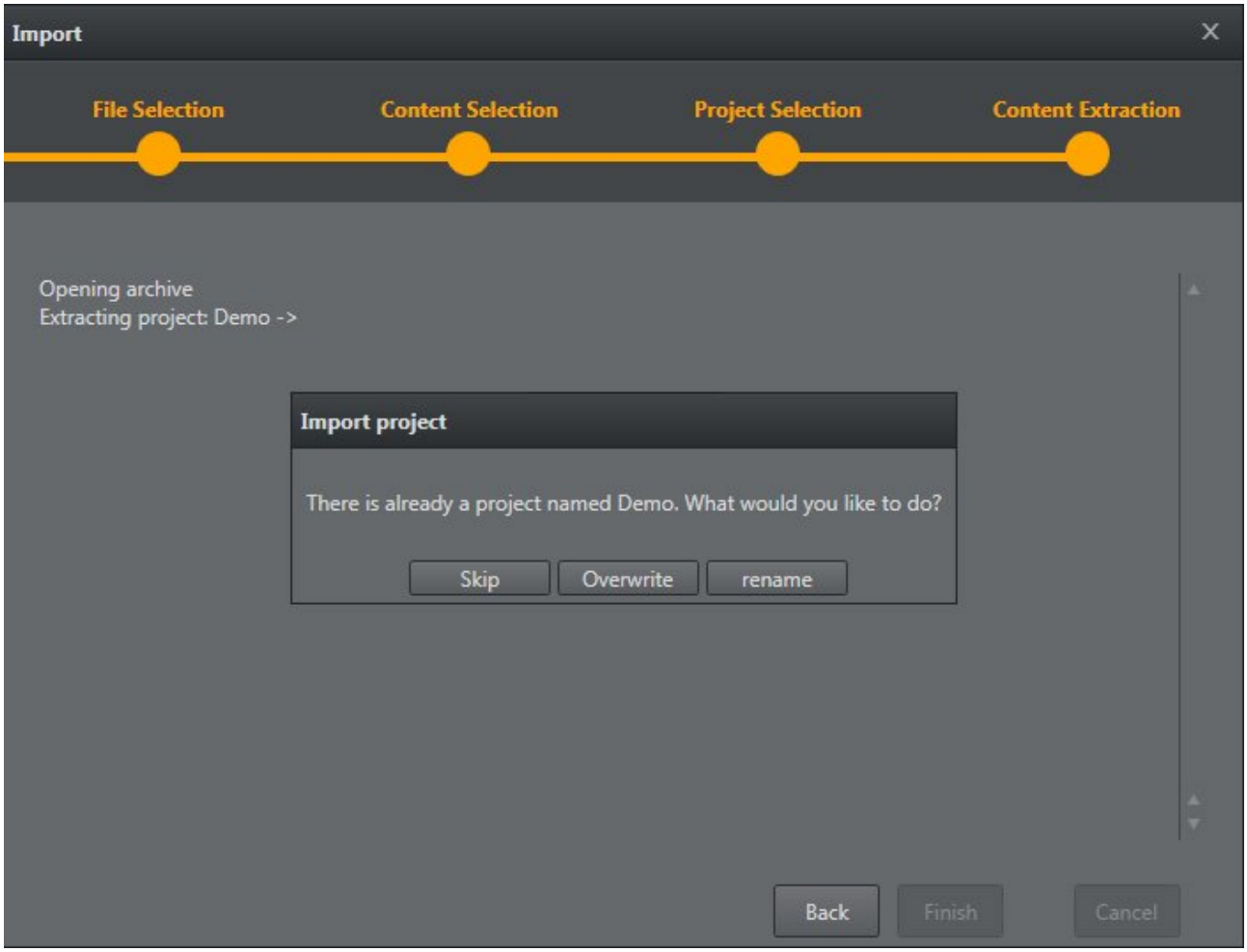
Select the file you wish to import.

Copyright © 2025 Vizrt



If the file already exists, you can skip the import process, rename the file or overwrite the file.

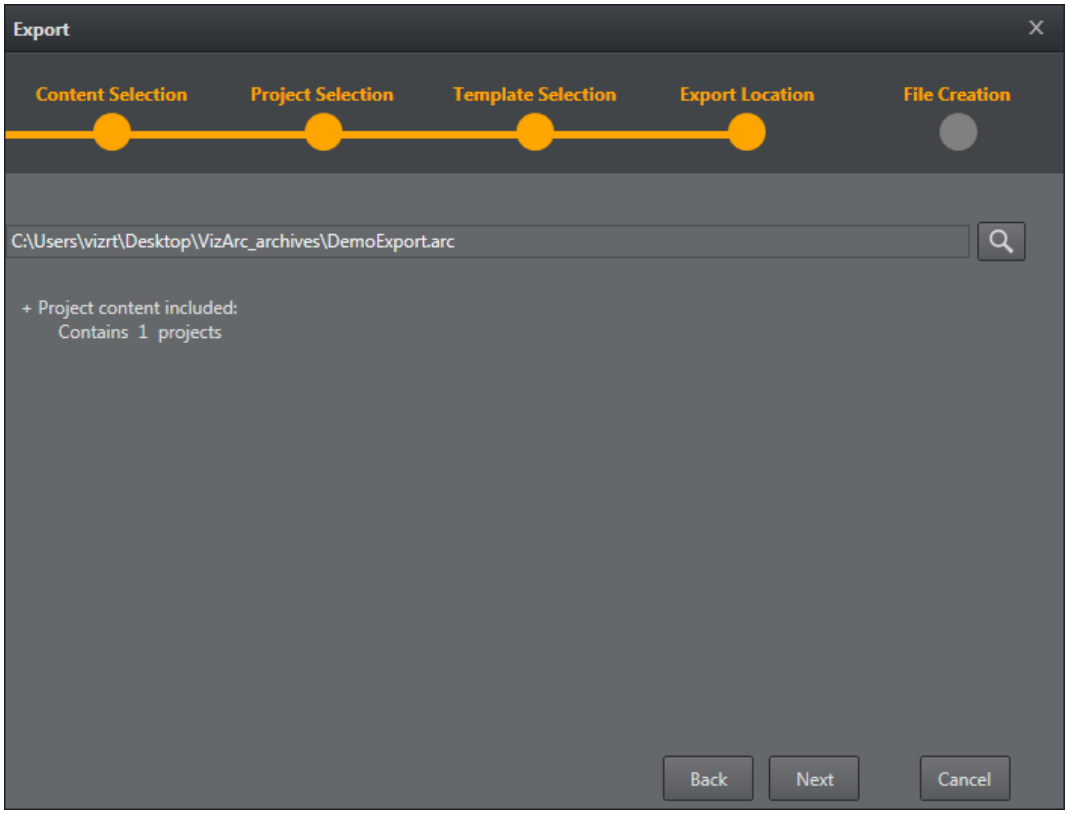
Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

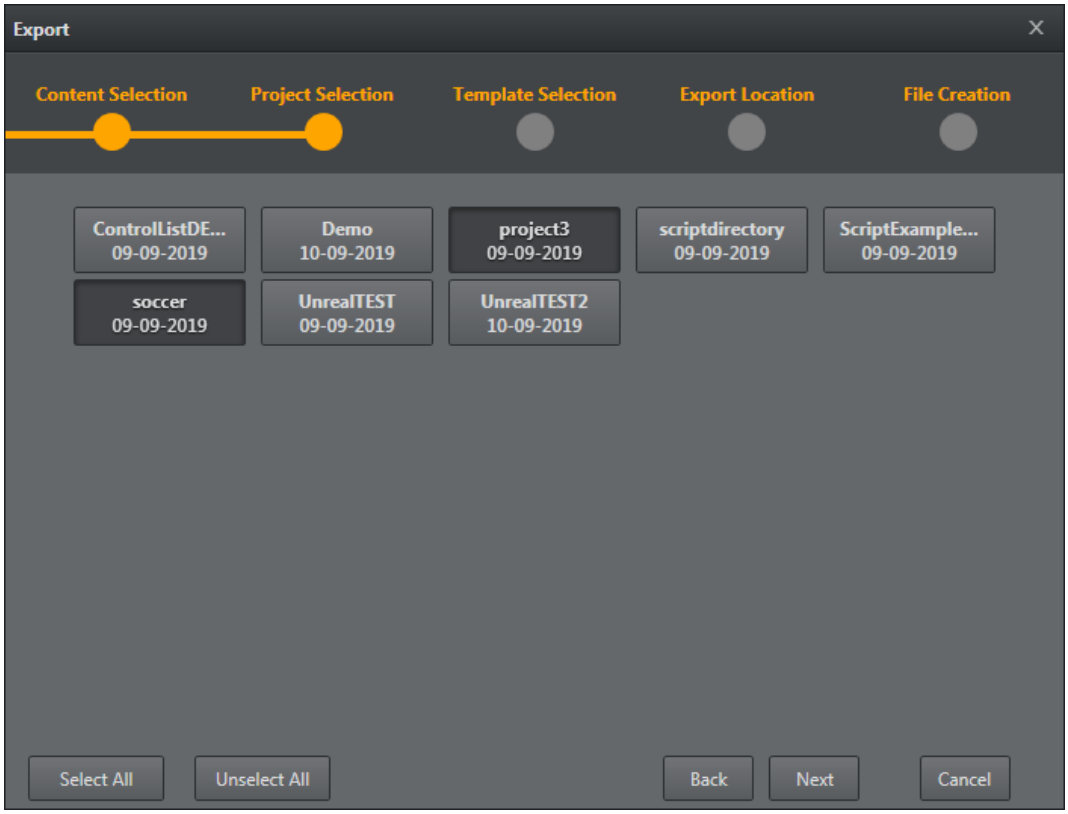
5.2.9 Export

Select **File > Export** to make the window below appear. Select which kind of file you want to export - **Projects**, **Profiles, Config** or **Keybind** and click **Next**.

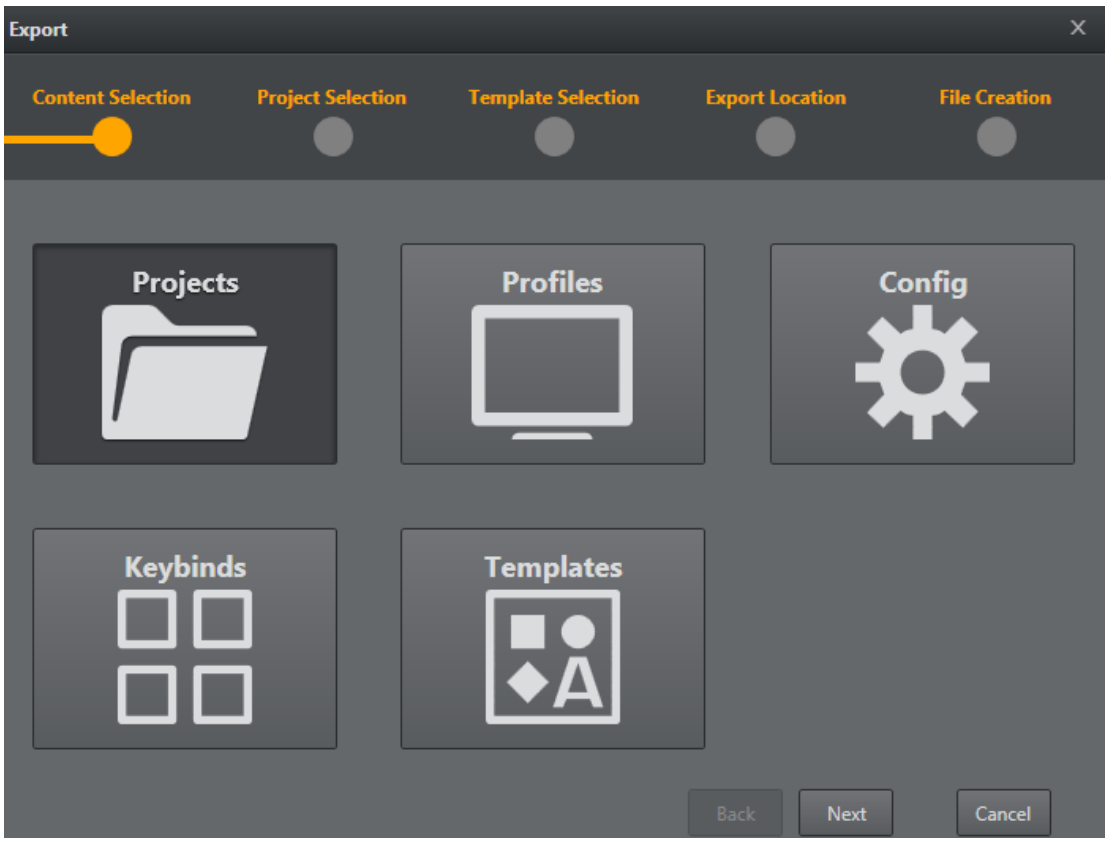


Select the file you wish to export.

Copyright © 2025 Vizrt

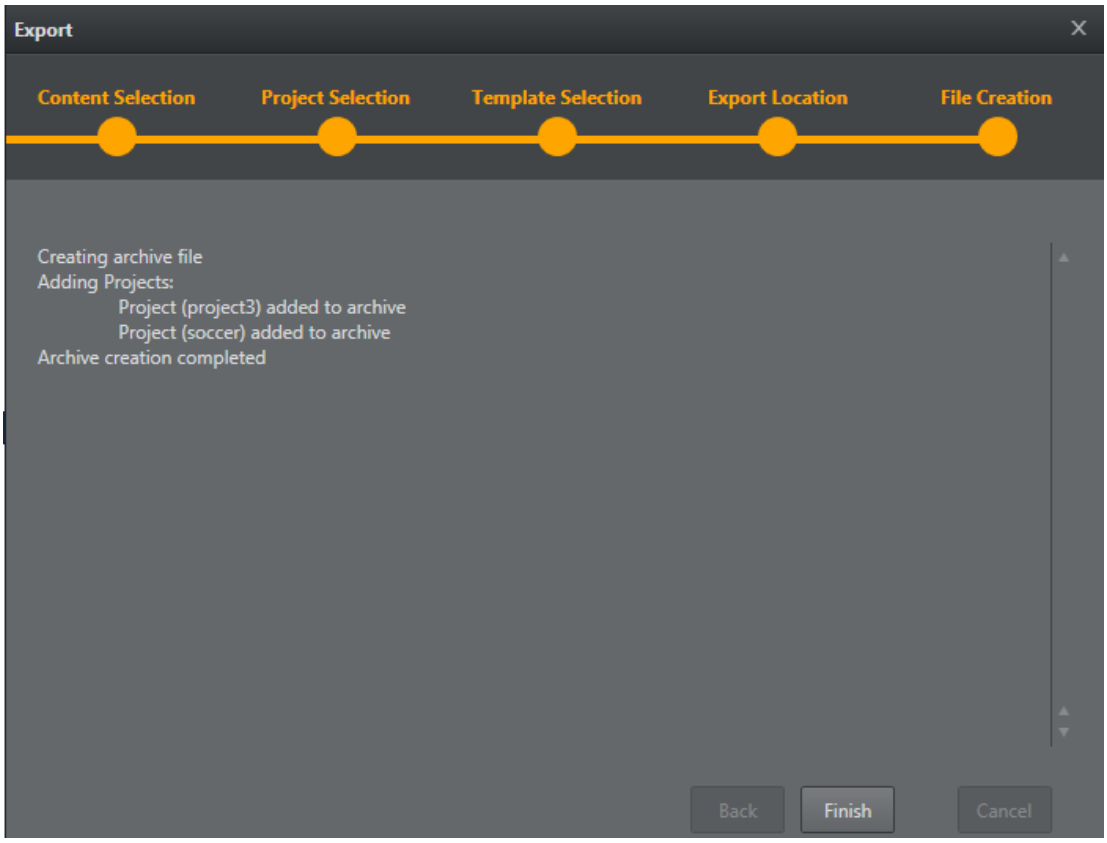


Select the file destination by clicking the search icon. A dialog appears. Click **Next**.



Copyright © 2025 Vizrt

Click the **Finish** button to complete the export action.

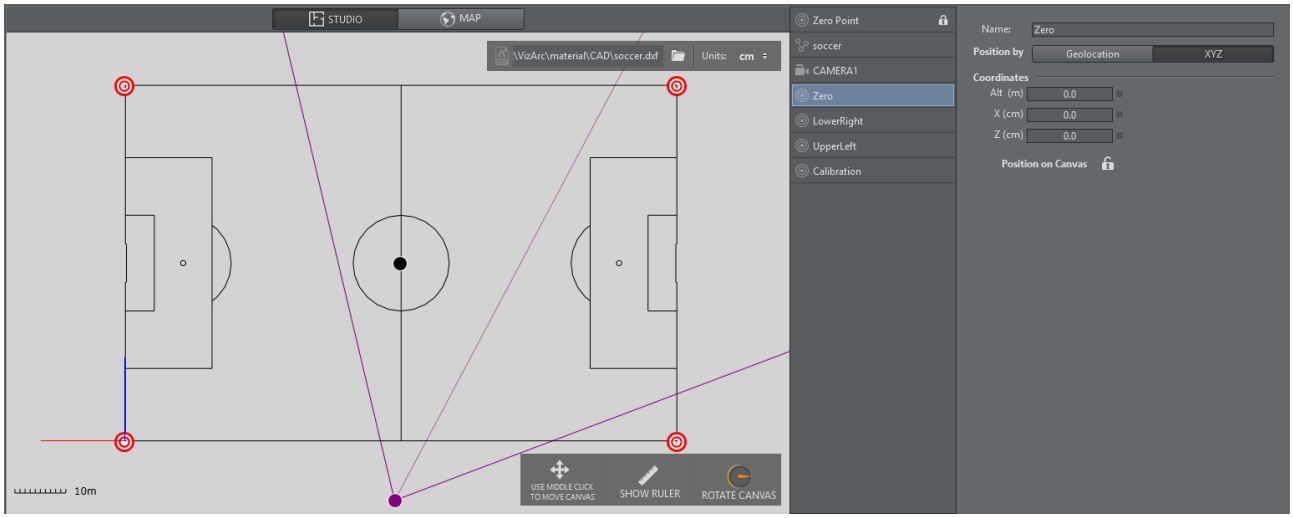


Copyright © 2025 Vizrt

5.3 4 Point Calibration

5.3.1 About 4 Point Calibration

4 Point Calibration can be used to calculate the position and orientation of a (typically mechanically) tracked camera. Using this type of calibration requires knowledge of at least four physical points in the domain (for example, the corners of a soccer pitch or well-known points on a horse race track).



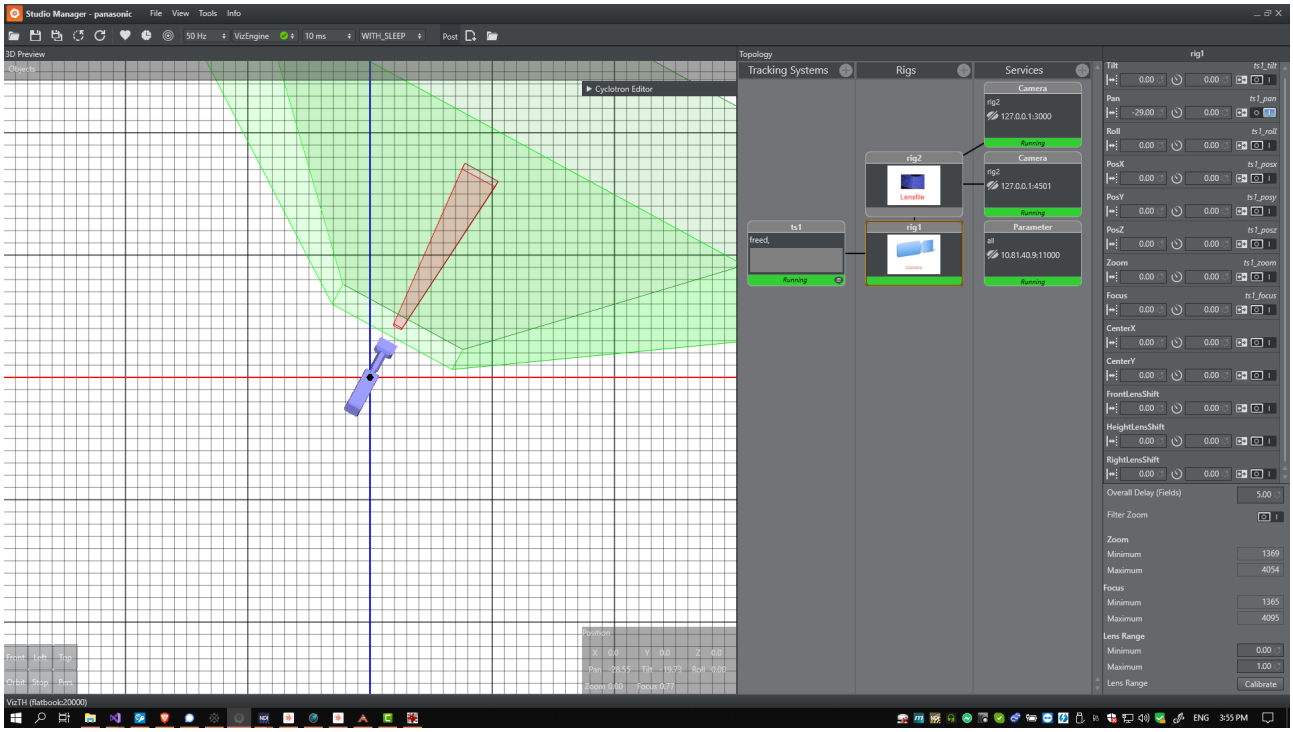
Map View showing a soccer pitch and four calibration points.

5.3.2 Prerequisites

- The calibration points must all be on the XZ plane (same height).
- The camera must be on a fixed position and can't be moved in space.
- At least four points need to be measured/known on the tracking domain.
- The points must define a surface and not just a one dimensional line.
- Make sure the tracking hub sends the tracking information to Viz Arc and make sure it includes the camera lens rig containing a valid calibration.

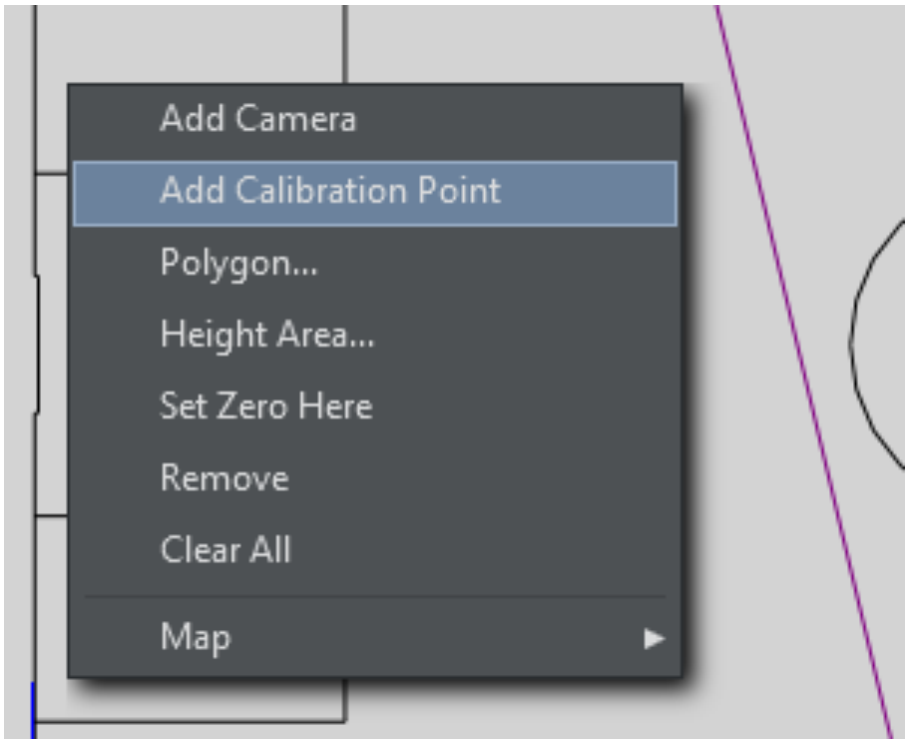
Note: Do NOT use a parent transformation rig on top for 4 Point calibration. The translation and rotation will be handled inside the root node of the Viz Artist scene.

Copyright © 2025 Vizrt



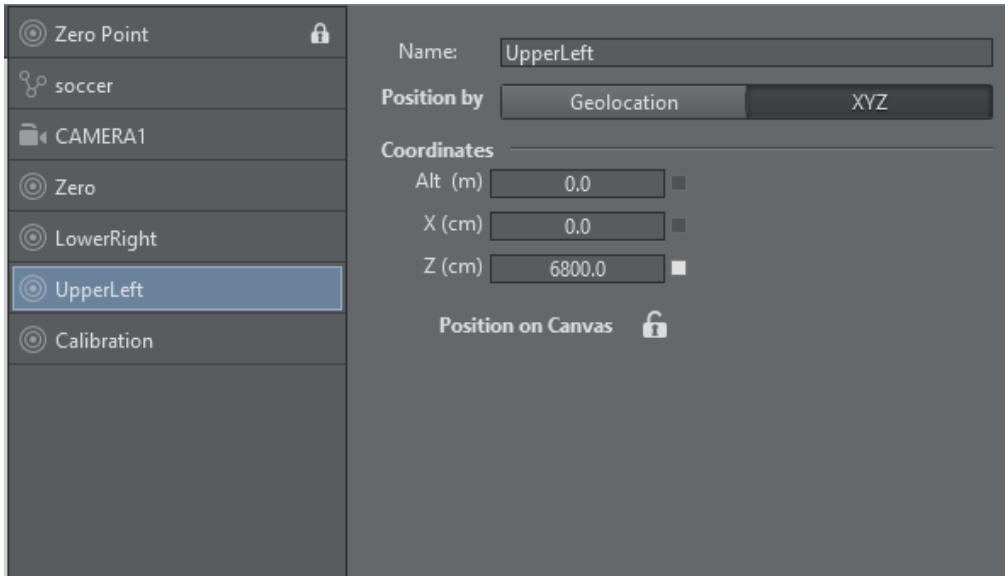
5.3.3 To Add Calibration Points

Add calibration points from the map's context menu:



Select the calibration point you just added and move it on the map, or enter its coordinates manually:

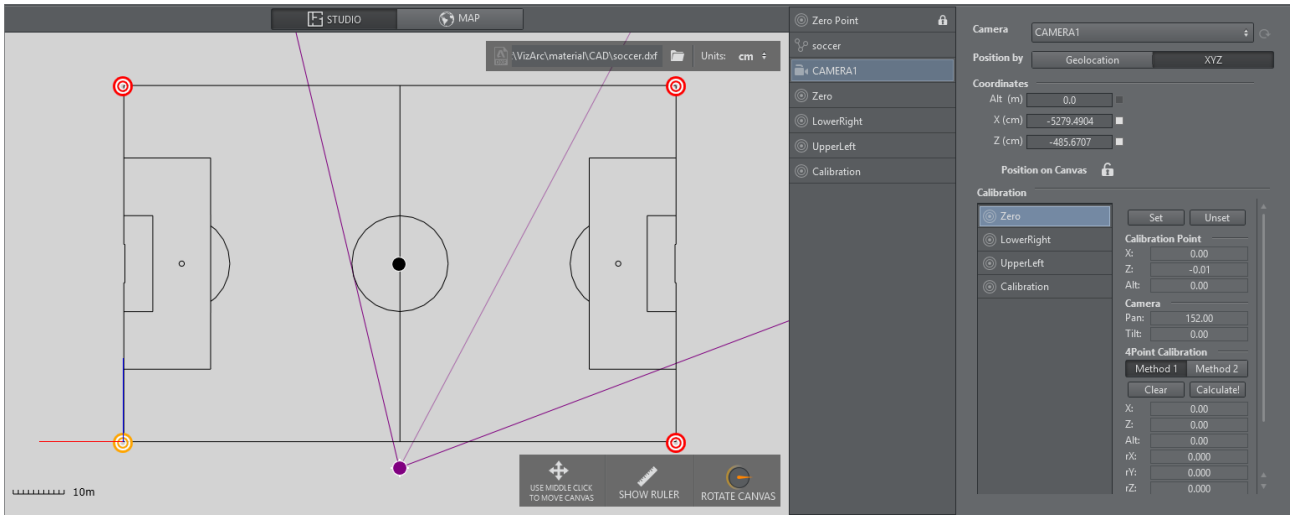
Copyright © 2025 Vizrt



Once all calibration points have been added and positioned, the actual calibration process may start.

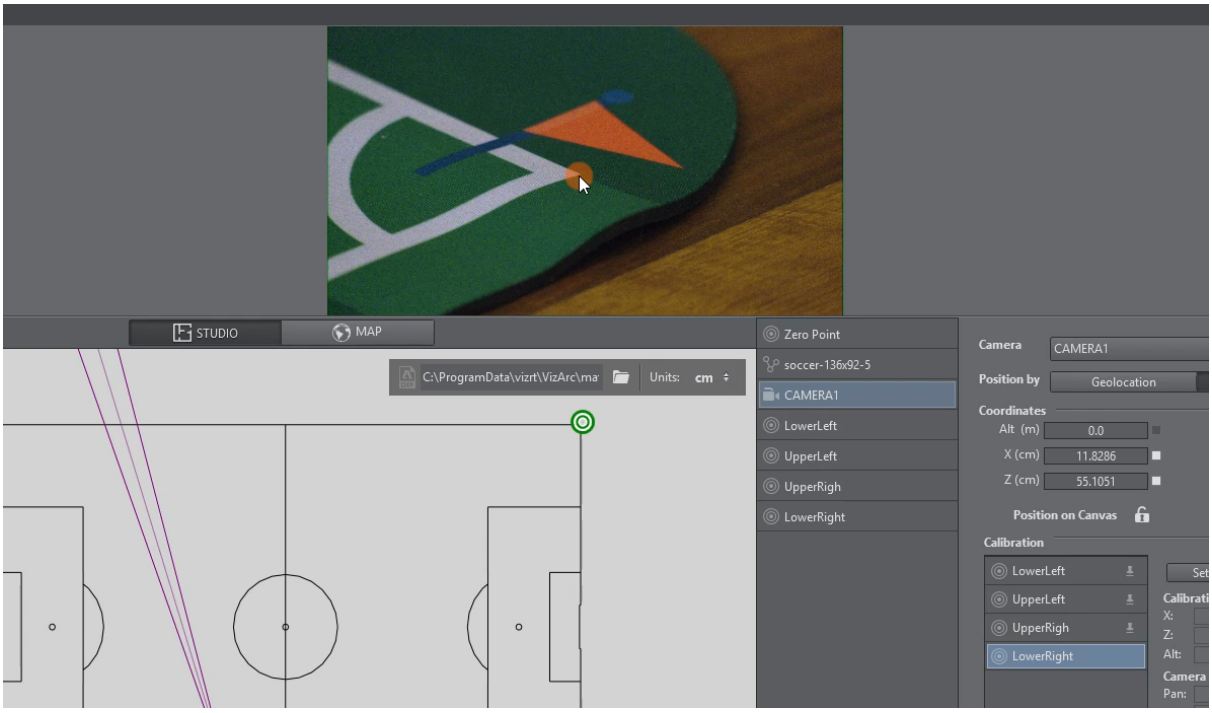
5.3.4 Calibration

Select the camera to be calibrated.



1. On the camera user interface select the marker towards which you want to point the camera. The calibration point is colored **yellow** on the map canvas.
2. The physical camera should now fully zoom into the marker and try to center it as best as possible.
3. Once the marker is centered, click the **Set** button. If your preview has been set up so that the camera is visible and you have a valid lens calibration, you can also click the position in the view instead of using the **Set** button.

Copyright © 2025 Vizrt

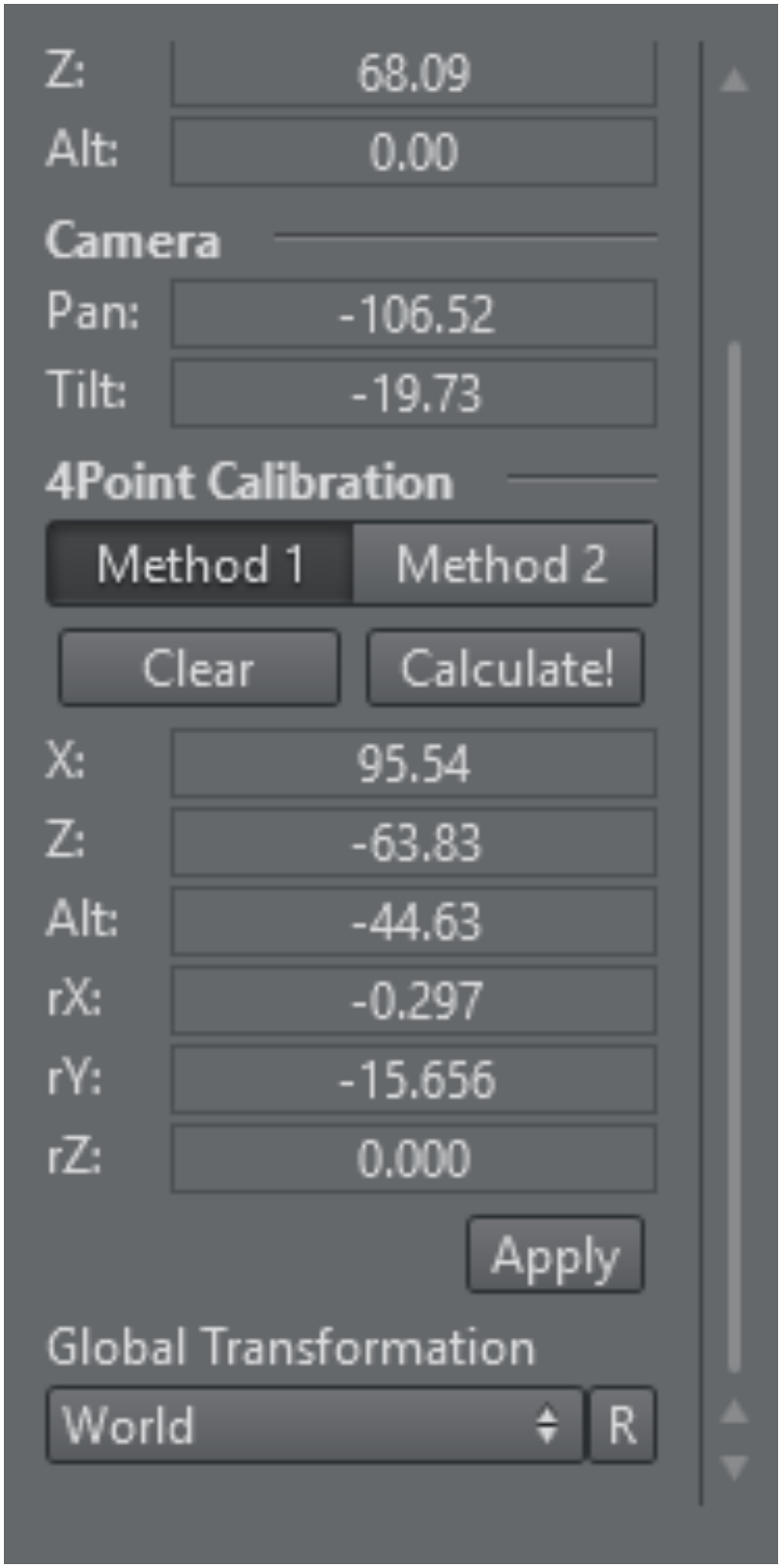


4. Continue this procedure for all or at least four calibration points.

5.3.5 Applying Calibration

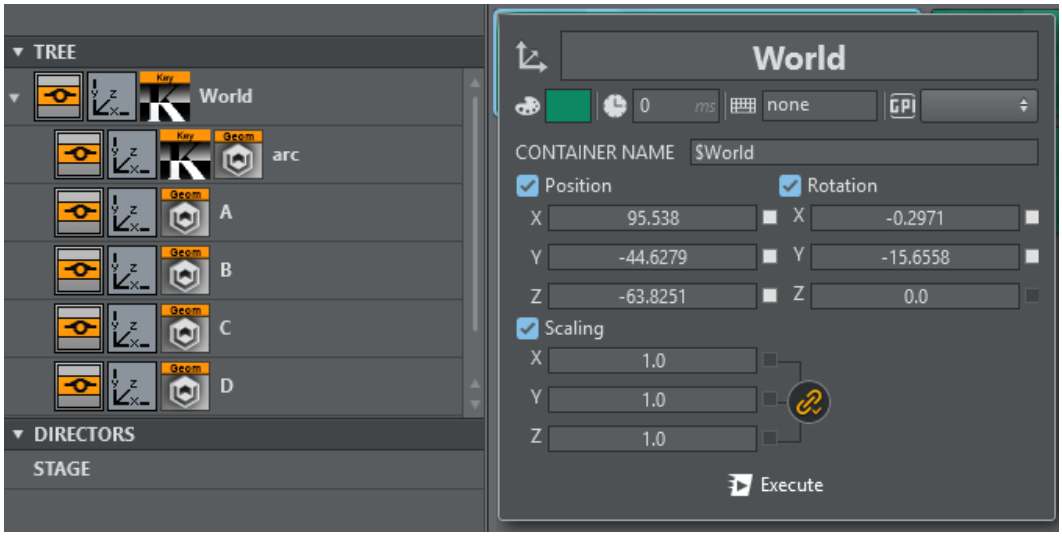
1. Once at least 4 points have been set, click the **Calculate!** button to calculate the presumed position and orientation of the camera. You can choose between **Method 1** and **Method 2**, hit **Calculate!** after switching algorithm. Generally Method 2 is more accurate but needs more and more accurate calibration points. Try both and pick the one that gives you better visual results.
2. Under the **Global Transformation** dropdown, select the transformation action holding the world transformation of the main scene (typically the root node of the AR/VR scene).
3. Click **Apply** to apply the calculated transformation and rotation.

Copyright © 2025 Vizrt

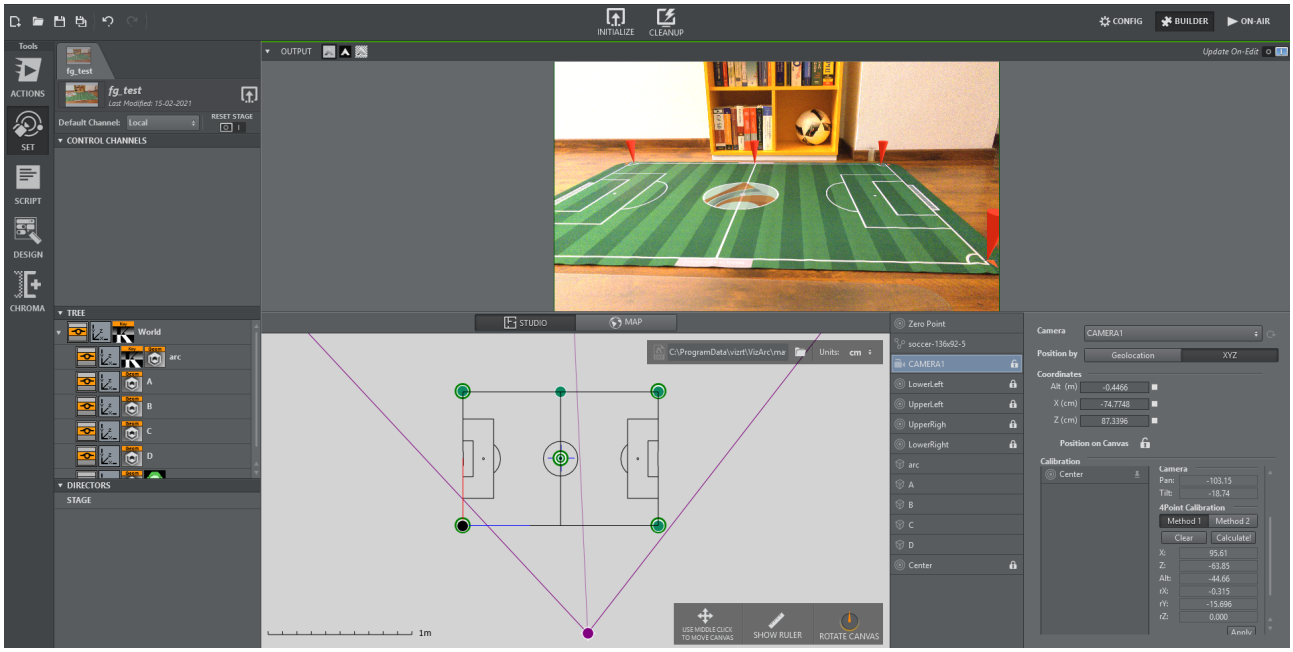


Use the **R** button to refresh the list of available transformation actions.

Copyright © 2025 Vizrt



Typically the transformation to be used is the root node of the scene.



Once calibrated you should be able to position AR graphics on your pitch.

Copyright © 2025 Vizrt

5.4 Supported Action Types

- Vizrt Scene Actions and Unreal Actions are created by dragging and dropping properties from scenes or level tree containers, such as geometry plugin (position, rotation, scaling), hide object action, or from a tree directly into the actions area.
- Project Actions are created by right-clicking on the actions canvas.
- Other Actions are created by right-clicking on an empty space in the main area.

Vizrt Scene Actions	Unreal Actions	Project Actions	Other Actions
Alpha	Active	Action Group	Flowics Playlist
ControlChannel	Animation	Chroma Key	Multizone Chroma Key
ControlObjects	Blueprint	Command	NDI
Director	Command	MSE	Scene Loader
Geometry	Dispatcher	Shared Memory	Tecnopoint Control
Image	Mesh	Tracking Hub	Telemetric Control
Key	Position/Rotation/ Scaling	Virtual Studio	Utah Router
Material	Sequence	Viz Camera	Vinten Control
Omo	Text		
PBR Material	Light		
Phong	Presets		
Script			
Text			
Transformation			
Viewport			
Visibility			

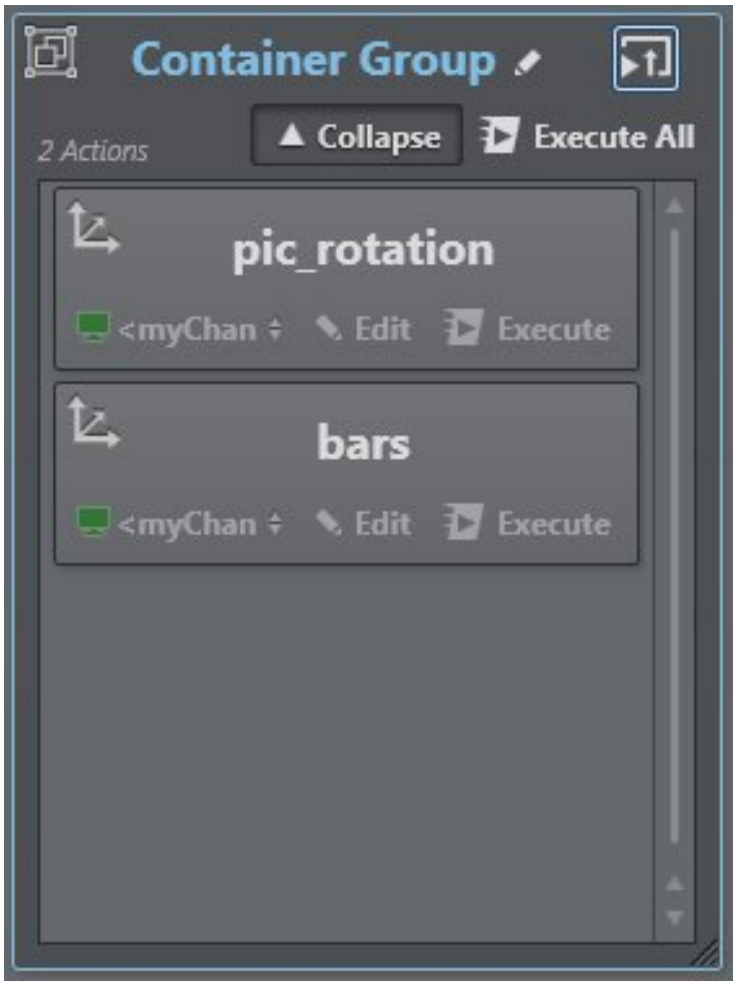
Copyright © 2025 Vizrt

5.4.1 Project Action Types



- Action Group (1)
- Command (2)
- Tracking Hub (3)
- MSE (4)
- Shared Memory (5)
- Chroma Key (6)
- Virtual Studio (7)
- Viz Camera (8)

Action Group

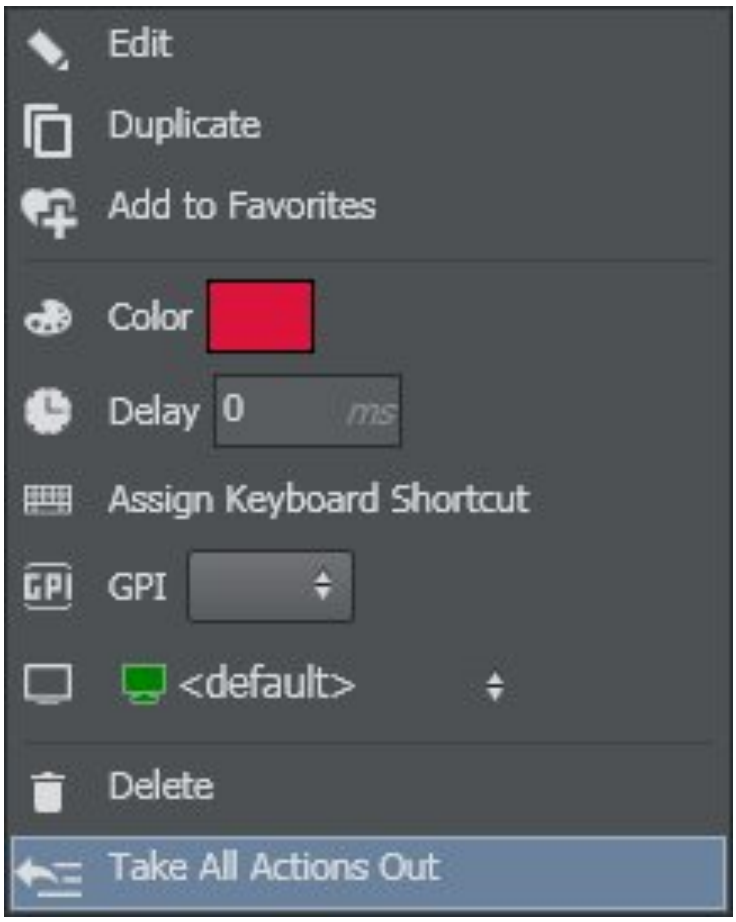


Action Group is useful for grouping several actions together, both scene and project, and executing them at the same time.
It's possible to collapse a group or expand it and insert/edit/remove action children.

Copyright © 2025 Vizrt

Pressing the **Edit** icon next to the group name opens the main tool, where you can edit the group name, box color, time delay in milliseconds, keyboard shortcut and GPI channel.

It's also possible to execute all action boxes included in an action group on initialize group by pressing the **Execute All** icon at the top-right of the group box.



Remove all actions from the group by selecting **Take All Actions Out** in the Context Menu (right-click on group action).

Copyright © 2025 Vizrt

Command



The Command action lets you write custom commands to talk with Viz Engines, for example to manage stage animations or start a specific director, as shown in the example above.

Tracking Hub

TH Command lets you write a custom command to interact with Tracking Hub.

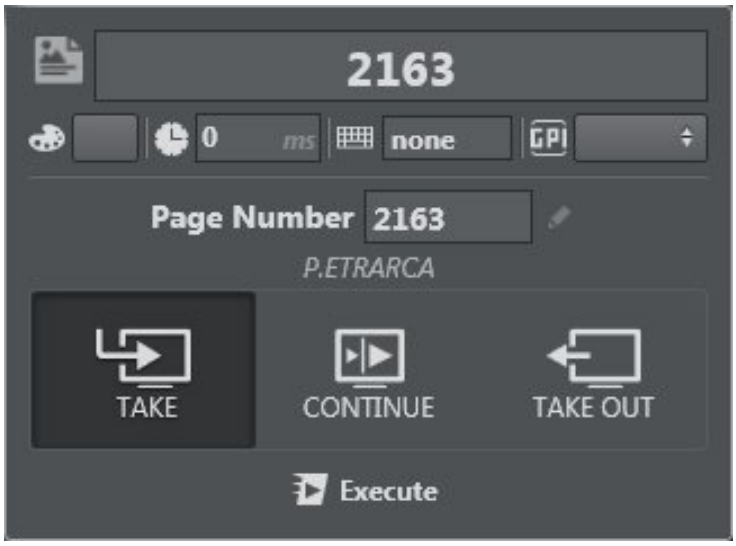


Note: Tracking Hub is a console program for Viz Virtual Studio that collects and stores data from tracking systems (cameras and objects). Viz Engine is provided with this tracking data at a configurable time or delay.

Copyright © 2025 Vizrt

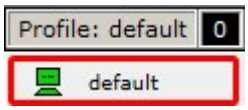
MSE





MSE lets you retrieve Viz Trio's saped pages and import them directly into Viz Arc for execution.



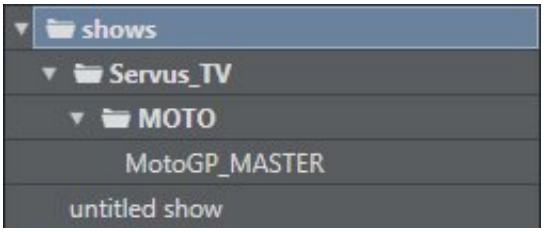


The profile used for the action is in the lower left corner of the MSE action. It contains a list of the profiles configured on your MSE.

To Import Trio Shows

This section shows you how to import Trio Saved Pages into Viz Arc:

Copyright © 2025 Vizrt



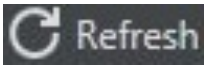
Press the **MSE icon** MSE to make a window containing a list of Trio shows appear.



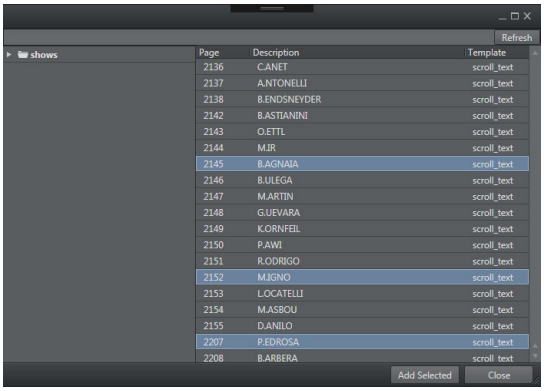
Page	Description	Template
1032	ANDREA SCHLAGER/	BB
1088	ANDREA SCHLAGER/@andrea.m.schlager/0	BB_FB_Twit
1027	ALEX HOFMANN & ANDREA SCHLAGER & GUSTL...	BB
1057	ALEX HOFMANN & ANDREA SCHLAGER/	BB
1014	ANDREA SCHLAGER & ALEX HOFMANN/	BB
1037	GUSTL AUINGER & ANDREA SCHLAGER/	BB
1147	ANDREA SCHLAGER & GUSTL AUINGER/	BB

Select a show. A list of Trio pages is displayed.

You can sort pages by page number, description and template by double-clicking on the header at the top of the list.

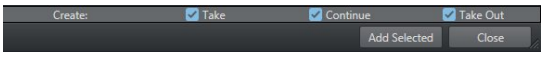


Reload MSE pages by clicking **Refresh**.



To import, select one or more pages and click **Add Selected** Add Selected to add them in the actions area.

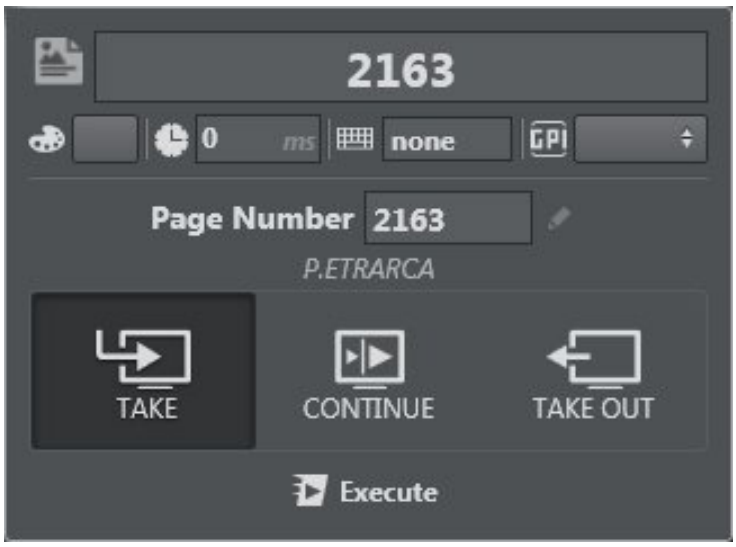
To abort an operation, close the window or press the **Close** Close button.



At the bottom of the MSE window, you can select **Take**, **Continue**, or **Take Out**, which creates a single action box for every operation.

Copyright © 2025 Vizrt

To Work with Page Animations

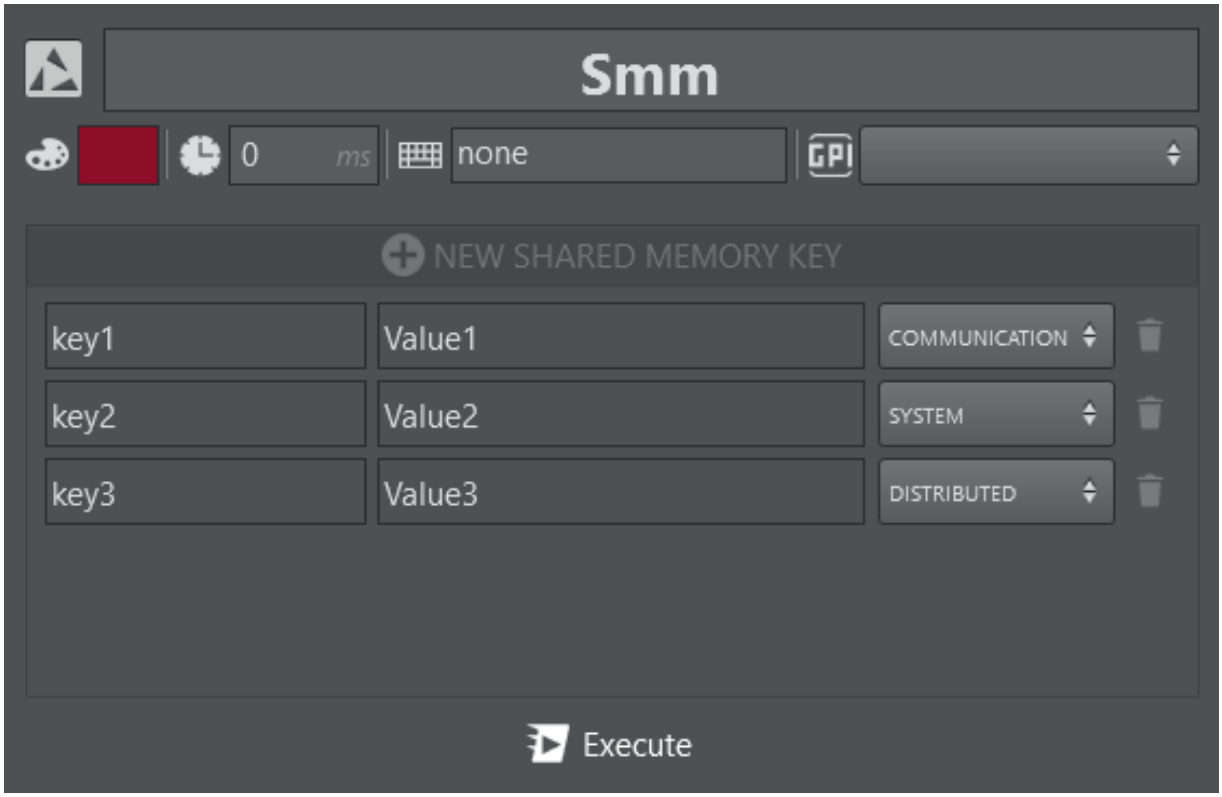


Once the pages are imported, you can work with Page animations:

	**TAKE:** start animation.
	**CONTINUE:** continue animation from the previous stop point to the next.
	**TAKE OUT:** take out the graphic.
	**Execute:** perform the action selected above.

Copyright © 2025 Vizrt

Shared Memory



Set Shared Memory names and assign a value using Shared Memory. You can add multiple Shared Memory key/value pairs.

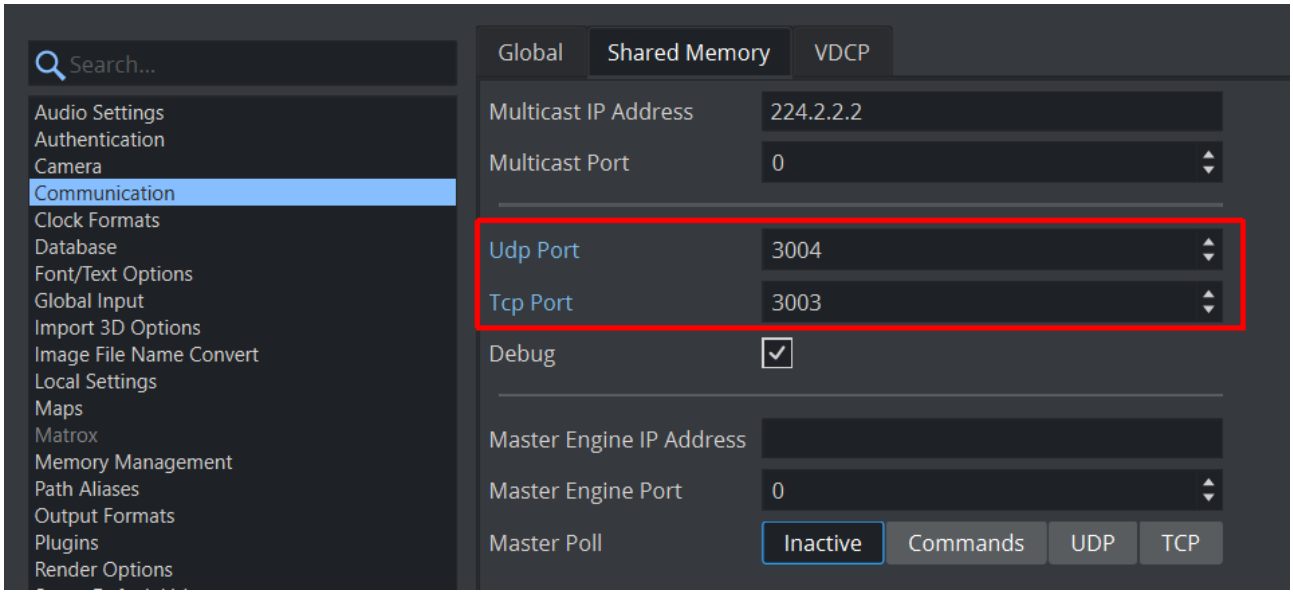
The value pairs can be sent to different shared memory destinations:

- **COMMUNICATION:** The VizCommunication map of the destination Engines is updated. Only the Engine's selected profile is updated.
- **SYSTEM:** The System map of the destination Engines is updated. Only the Engine's selected profile is updated.
- **DISTRIBUTED:** The VizCommunication map of the destination Engines is updated and all Engines connected with the same user to the Graphic Hub are updated as well.

The action works only when the destination Engine(s) have either UDP or TCP ports configured for shared memory updates. It uses the UDP port if configured other than `0`, else it uses the TCP if configured other than `0`.

The Viz Engine configuration, configured to receive shared memory updates via UDP on port `3004` or via TCP on port `3003`.

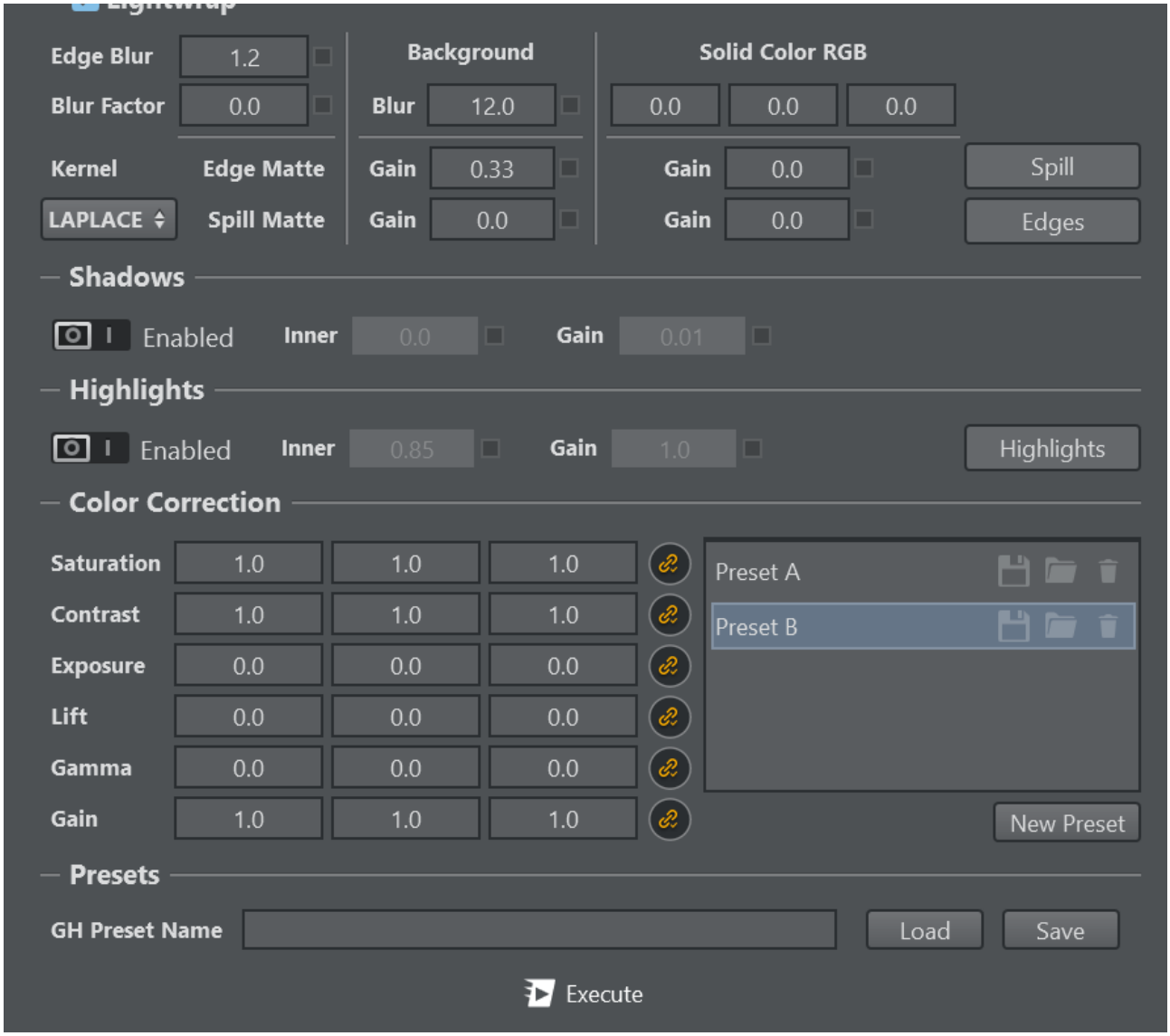
Copyright © 2025 Vizrt



Precision Keyer



Copyright © 2025 Vizrt



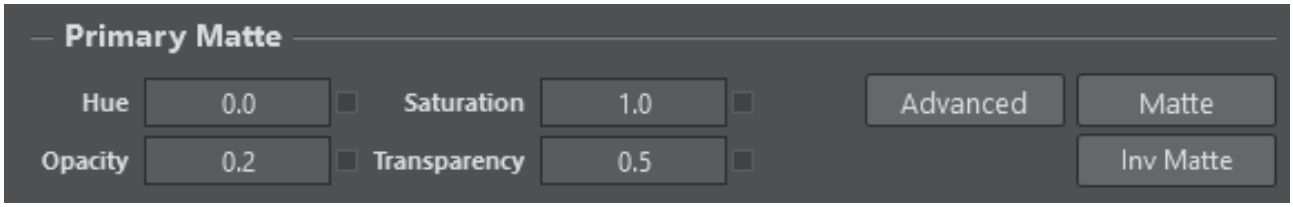
Setting up the Precision Keyer is a sequential process where each step creates an interim result. The suggested layout follows the sequence and places the show buttons next to the items which need to be inspected. The buttons shown are check boxes and can be overwritten if another checkbox is selected. If no checkbox is active, the final composition is shown.

No checkbox	Shows the final and composed output of the Keyer.
Trash Matte	Shows the color coded trash matte on the output.
Matte	Shows the key output, white representing foreground and black the background.
Matte Inverse	Shows the inverse of the Matte output, foreground white, background black.

Copyright © 2025 Vizrt

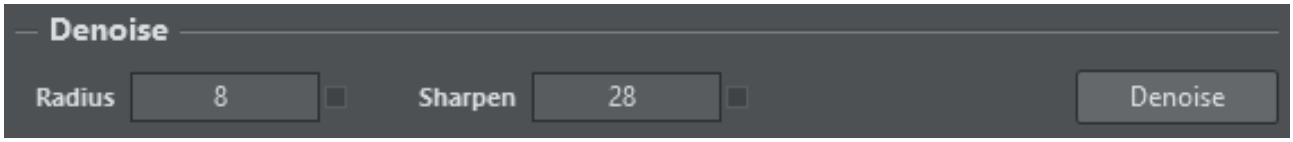
Advanced (Primary Matte)	Inks threshold values to ease setup when it is hard to distinguish between 0.9 and 1.0 (both appear "white" on monitors). Value 0.0 is mapped to green and 1.0 to red.
Denoise	Shows the output of the de-noising step.
Processed	Shows the processed output.
Advanced (Processed Foreground)	Inks threshold values to ease setup when it is hard to distinguish between 0.9 and 1.0 (both appear "white" on monitors). Value 0.0 (fully processed) is mapped to red.
Spill	Shows the spill matte.
Edges	Shows the detected edges between foreground and background.
Highlights	Shows the highlights in the foreground.

Primary Matte



- **Hue:** Rotate the color space to move the color of the Cyclorama to the purest color chosen, green or blue. You can adjust this by looking at the matte and checking that the matte calculated on the Cyclorama approaches black.
- **Saturation:** If the video is not highly saturated you may want to adjust this to ease setting of the opacity and transparency point.
- **Opacity:** This is the point at which the primary matte value becomes 100% opaque. Equivalent to the black point on gray scale images.
- **Transparency:** This is the point at the which the primary matte becomes 100% transparent. Equivalent to the white point in gray scale images.

Matte Denoise

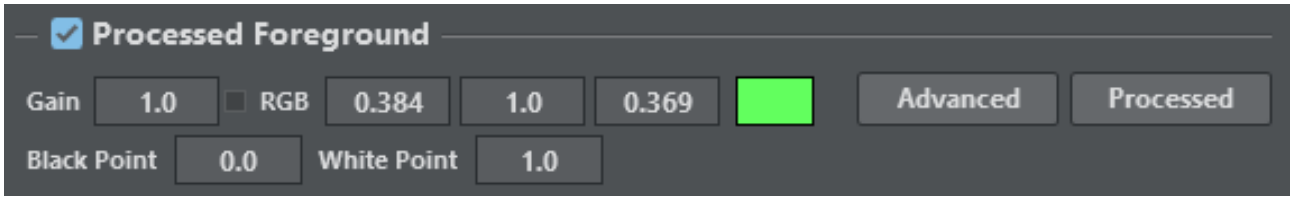


- **Radius:** The radius of the bi-linear filter used to blur the noisy parts of the image.

Copyright © 2025 Vizrt

- **Sharpen:** Pixels whose color differ more than this threshold from the main pixel are not included in the blurring process. This maintains detail, even if a large blur radius is applied.

Processed Foreground



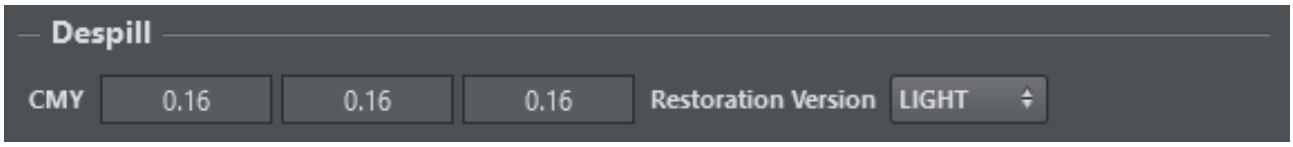
- **Process Foreground:** Enables or disables the foreground processing. When disabled, it falls back to a classic alpha blending.
- **Gain:** Sets gain for the processed foreground color.
- **RGB:** Uses this backing plate color to process the foreground. This value gets subtracted from the foreground color, and the result should be black.
- **Black Point:** Scales the key value used when processing the foreground. Only affects areas where matte is > 0.0 and < 1.0. Can be used to enlarge the talent in semi-transparent regions, increases lightness around edges.
- **White Point:** Scales the key used when processing the foreground. Only affects areas where matte is > 0.0 and < 1.0. Can be used to shrink the talent in semi-transparent regions, decreases lightness around edges.

The Black Point can be used to enlarge talent, the White Point to shrink it. It only affects foreground processing and transition regions. Shrunken talent introduces darkened edges, enlarged talent enlightens edges.

How to setup RGB values manually:

- Enabled Advanced mode, it highlights areas which are fully processed in RED (when processed - color is less than 0).
- Turn all values to 1.0, background should be fully red now, increase gain if it's not.
- Tune in B: Reduce value until the red areas start to vanish. Go back a little bit.
- Tune in G: Reduce value until the red areas start to vanish. Go back a little bit.
- Tune in R: Reduce value until the red areas start to vanish. Go back a little bit.
- Switch back to normal mode.
- Check result around edges, fine tune as necessary.

Despill



- **CMY:** Cyan, Magenta and Yellow despill gain.
- **Restoration Version:** Restores more lightness during despilling when LIGHT is selected.

Copyright © 2025 Vizrt

Lightwrap



Light wrapping: Light wrapping is a compositing technique designed to blend keyed out green screen footage with a background plate. Light wrapping is used to simulate ambient light that casts light onto your subject in the scene. Although in some circumstances (like mixed lighting) it can be better to use a different layer than your background layer for the light wrap, this all depends on your scene.

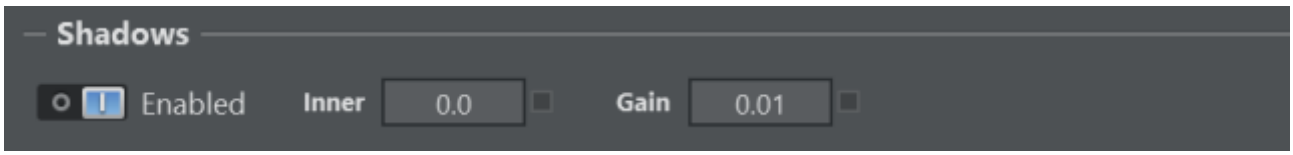
- **Lightwrap:** Enables or disables lightwrapping.
- **Edge Blur:** If this value is greater than zero, edge detection is performed and edges found are blurred by the amount set in this field. The blurred edges serve as input to brighten up the edges to overcome overshooting when going from a dark color to the Cyclorama. These edges are used for lightwrapping.
- **Edge Blur Factor:** Determines how much of the background blur is used for the edge lightwrapping. 1.0 = same blur as for spill lightwrapping, 0.0 = use unblurred background.
- **Kernel:** Choose between LAPLACE and SOBEL. LAPLACE is more fine-grained and keeps more details.
- **Blur:** The blurred background is used for lightwrapping onto the foreground together with the spill matte or the edge matte. See also Edge Blur Factor.
- **Edge Gain - Background:** Defines how strongly the lightwrap is applied to the background when using the edge matte.
- **Spill Gain - Background:** Defines how strongly the lightwrap is applied to the background when using the spill matte.
- **Solid Color RGB:** A constant color that is applied as a lightwrap on the foreground.
- **Edge Gain - Solid Color RGB:** Defines how strongly the lightwrap is applied to the foreground when using the edge matte.
- **Spill Gain - Solid Color RGB:** Defines how strongly the lightwrap is applied to the foreground when using the spill matte.

Spill: Carefully use **Spill Gain - Background** to create an ambient light effect, and increase **Edge Gain - Background** to improve edge quality. Blur level should be high to prevent transparency effects when using **Spill Gain**, while **Edge Blur Factor** can be low to reduce blur used for edge lightwrapping.

Edge Matte: Edge detection is performed on the primary matte, resulting in a grayscale image that in most cases needs to be blurred for a smoother transition.

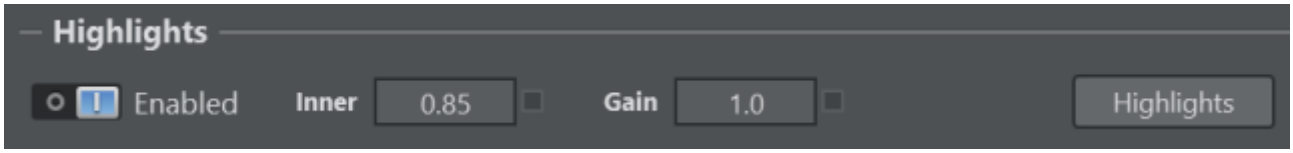
Copyright © 2025 Vizrt

Shadows



- **Enabled:** Toggles background shadows.
- **Inner:** Sets threshold under which shadows or dark colors are brought to the background. This feature is disabled if the value is zero.
- **Gain:** Strengthens the effect of bringing dark colors back.

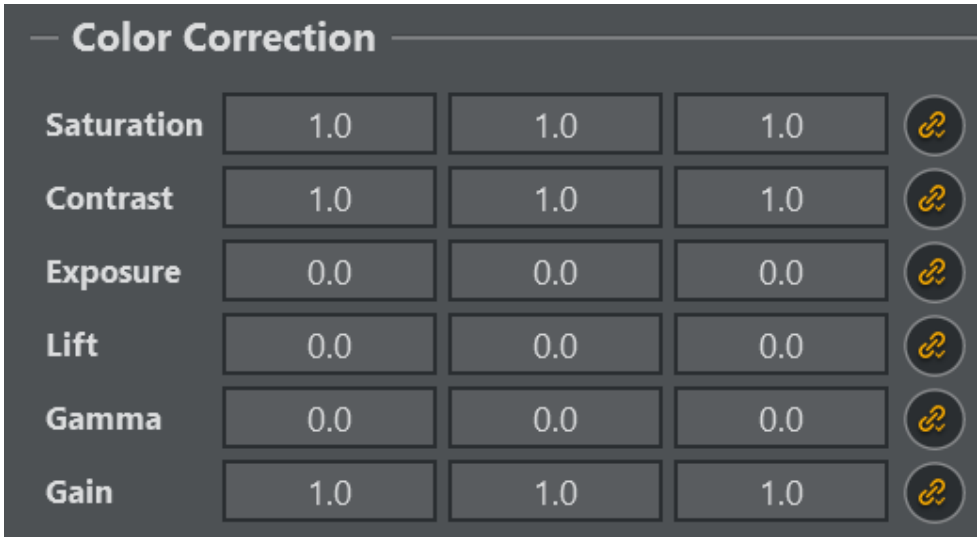
Highlights



- **Enabled:** Toggles foreground highlights.
- **Inner:** Luminescence from which highlights are added to the background.
- **Gain:** Highlight gain.

Color Correction

Primary color correction is applied on the already keyed foreground pixels.



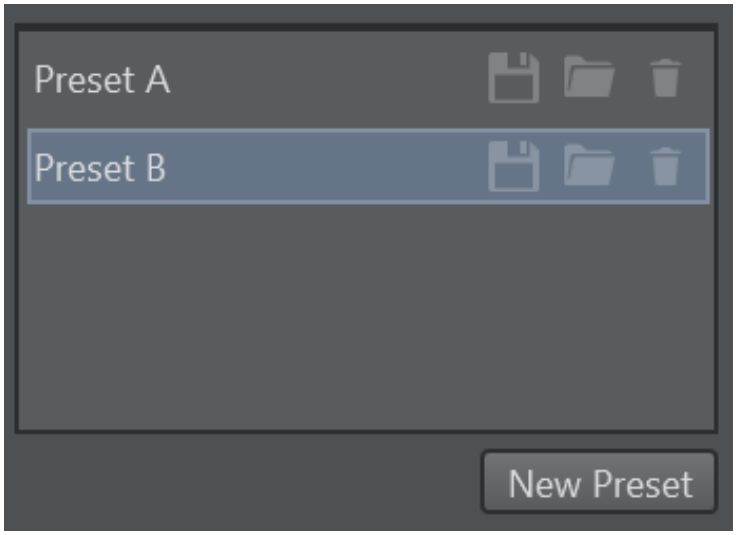
- **Saturation:** Values for the overall saturation.
- **Contrast:** Values for the overall contrast.
- **Exposure:** Values for the overall exposure.
- **Lift:** RGB values for lift.
- **Gamma:** RGB values for gamma.

Copyright © 2025 Vizrt

- **Gain:** RGB values for gain.

Presets

Presets can be used to store and recall different presets within the chroma action. Its main purpose is to be able to quickly switch between stored presets and experiment with some settings while being able to quickly undo the changes recalling a stored preset.



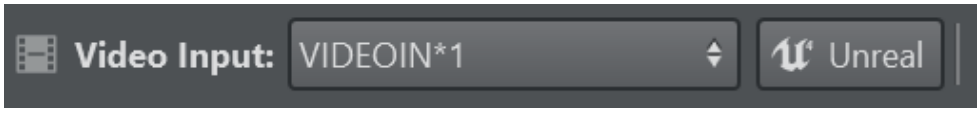
Create a new preset by clicking the **New Preset** button. It can be renamed using the **Rename...** context menu entry. When a new preset is created the current chroma keying values are stored. After changing some values use the save icon to overwrite the respective preset.

Recall a preset by double clicking the preset entry or by clicking the load icon. Delete a preset using the trash icon or by selecting the **Delete...** context menu.

> **Note:** The save buttons in the preset stores the preset within the Precision Keyer action. To store them permanently on disk, save the entire project on the local file system or the Graphic Hub.

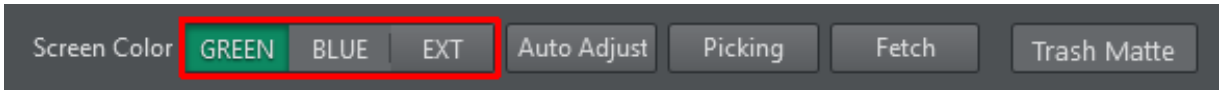
Video Input

Select the asset source to apply the keying on. Toggle the **Unreal** button to specify that keying takes place in the Unreal Engine.



Screen Color

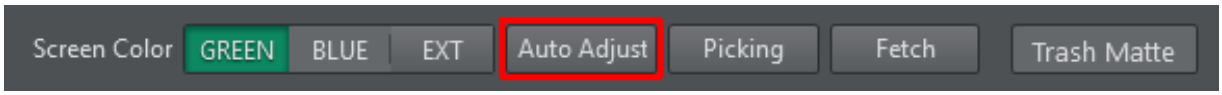
Select between **GREEN** and **BLUE** screen color, or **EXT** in order use key and processed foreground from external input, skips internal matte calculation and despiller.



Copyright © 2025 Vizrt

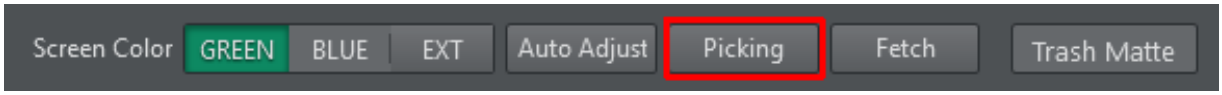
Auto Adjust

For an initial keying setup, point the camera into the green/blue box and hit the **Auto Adjust** button to get an initial key.

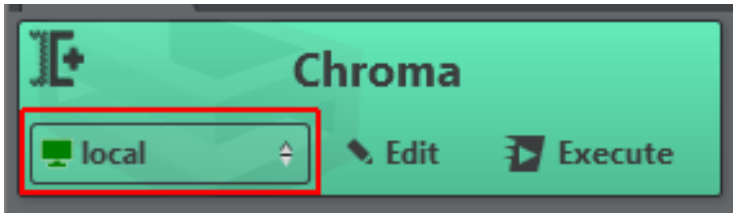


Picking

Picking lets you get a good first estimate of your Chroma/Precision Keyer. Press the **Picking** button to enter picking mode:

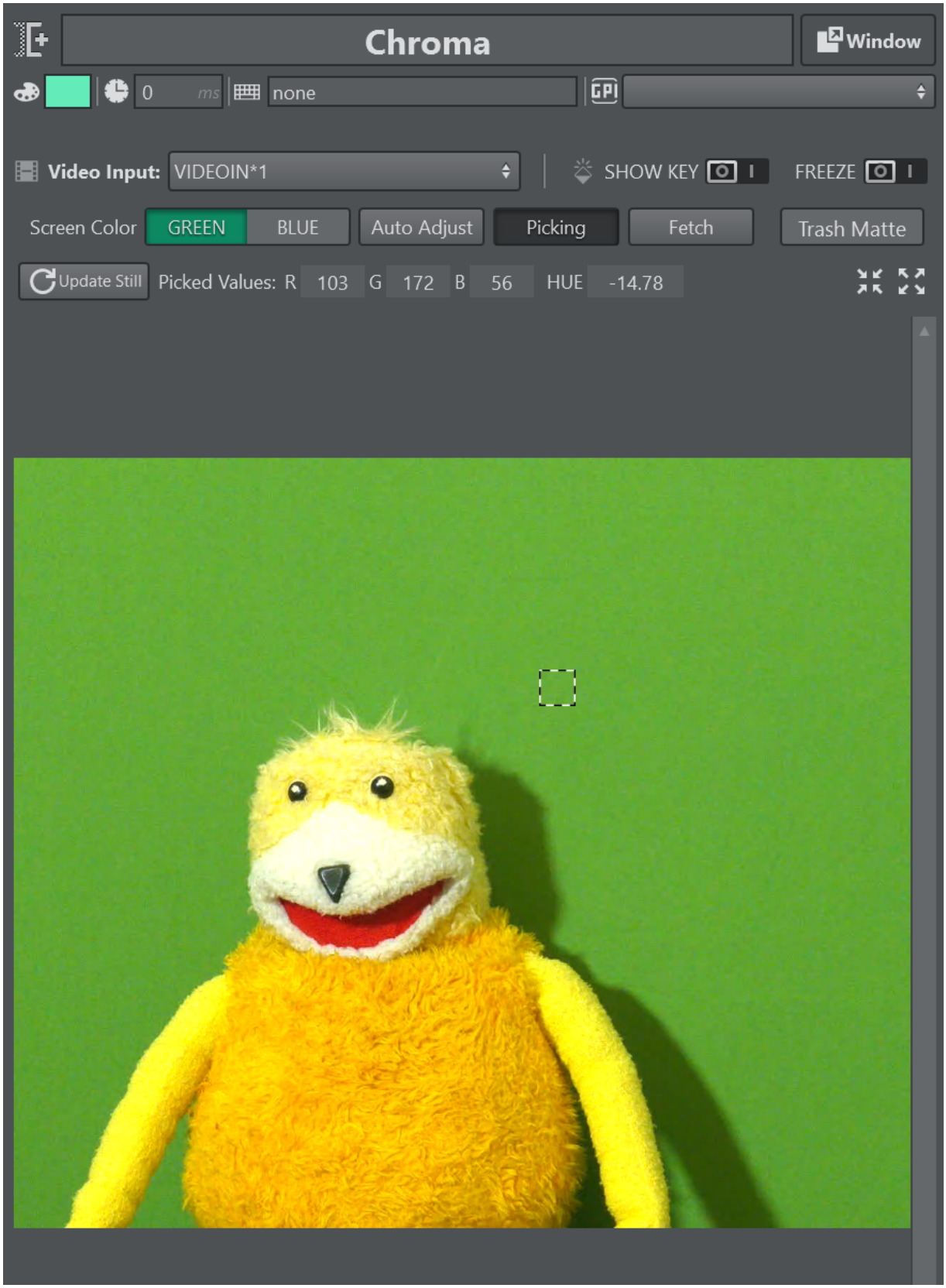


An attempt is automatically made to fetch a still frame from Viz Engine configured in the current channel selected in the chroma action.



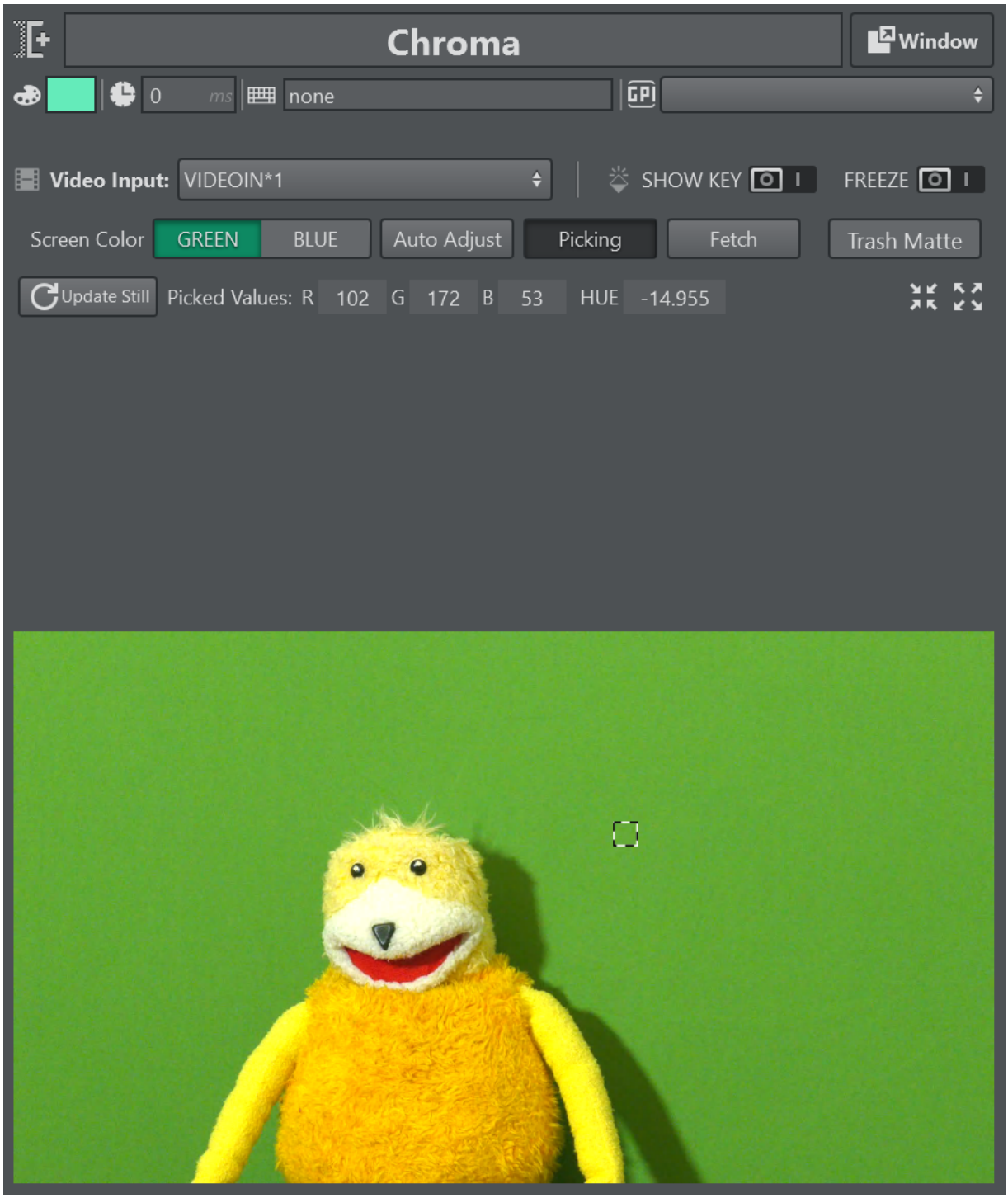
You'll now be able to scroll horizontally and vertically within the snapshot and pick an area of green. The picking area can be resized using the mouse wheel.

Copyright © 2025 Vizrt



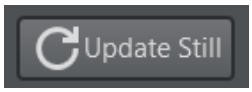
Copyright © 2025 Vizrt

Once an area is picked, the calculated RGB values are displayed above the snapshot. The **HUE** value is only relevant if the Precision Keyer is selected. The picked values are directly applied to the Engine if *Update On-Edit* is enabled.

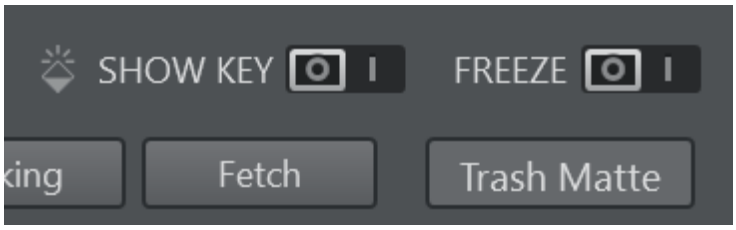




You can use the **Fit** and **Scroll** to view the snapshot either scaled into the view or in it's original size for more accurate picking.



To fetch a new still frame from Viz Engine for picking, press **Update Still**.



- The **Fetch** button retrieves the current keying settings from the Viz Engine. This is useful when creating a new Chroma Action that needs to be initialized with the current settings of Viz Engine.
- **FREEZE** freezes the selected video input on the Engine. Don't forget to unfreeze when you are finished adjusting the key.
- **SHOW KEY** forces the Viz Engine to output the key channel on the fill output. Deselect this when you are finished adjusting the key.

Adjusting the Keyer

To get to a good keying result, the following steps should be followed in the described order, since the different steps depend on each other.

- Start by hitting the **Auto Adjust** key button in the Keyer control. This sets the values of the hue and saturation and ensures that the rest of the keying parameters are set to a value to optimize their influence on previous settings.
- The obtained result after hitting the **Auto Adjust** key button is the primary matte which results from the difference between the pixel colors and the primary color. The ideal hue can be found by viewing the matte and rotating the hue until the key strength is at a maximum. The brightness of the key represents the strength of the key. The saturation can be used to increase the key strength in case the signal coming from the camera is not very saturated. To adjust the limits for transparency and opacity the user can switch between the key and inverse key view. Clicking on the same button again disables the view.
- Denoising can be viewed either on the key signal or on the denoised frame which is used to calculate the key. A large radius has a performance cost. Denoising should eliminate the noise on homogenous areas, while keeping details like hair intact.
- The processed foreground is the next step. The background should be replaced by solid black. When reducing the gain, the user can check if too much or not enough red or blue is still in the processed foreground. Subtracting the wrong color results in red or blue tint in areas with semi-transparent key.
- Despilling serves two purposes: It removes the green or blue spill from the people and objects in the studio and generates a matte which can be used for light wrapping with either a solid color or blurred background. The shadows and highlights sections allow to bring back dark and bright colors. This happens after the processed foreground has been applied to the background. This should be done after adjusting the primary key and processed foreground.
- Lightwrapping helps to blend foreground and background better and should be the last step in the process of keying/compositing.

Copyright © 2025 Vizrt

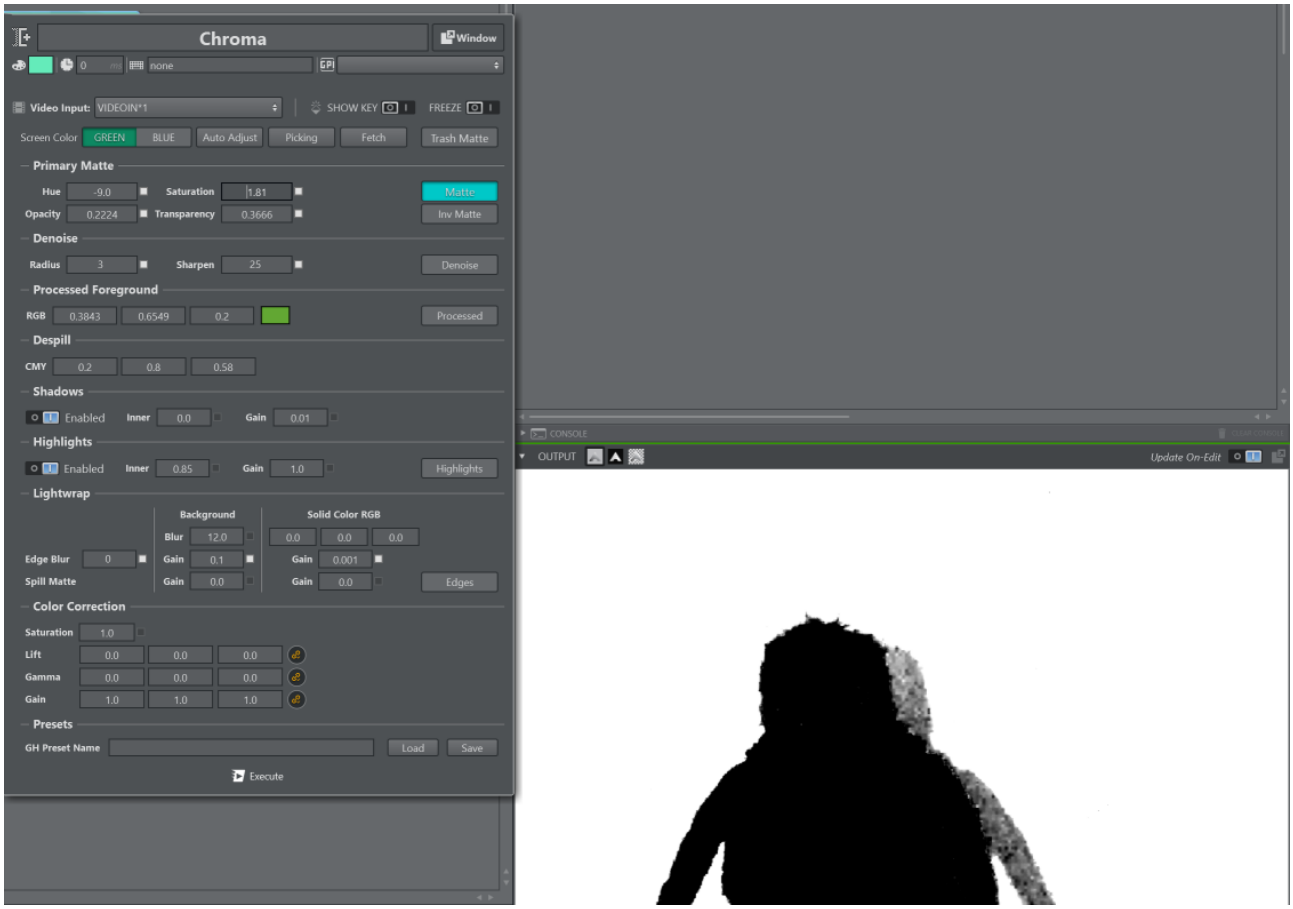
- A color correction can be applied to the processed foreground. It has no effect on any of the keying and blending methods.

Precision Keyer Stages



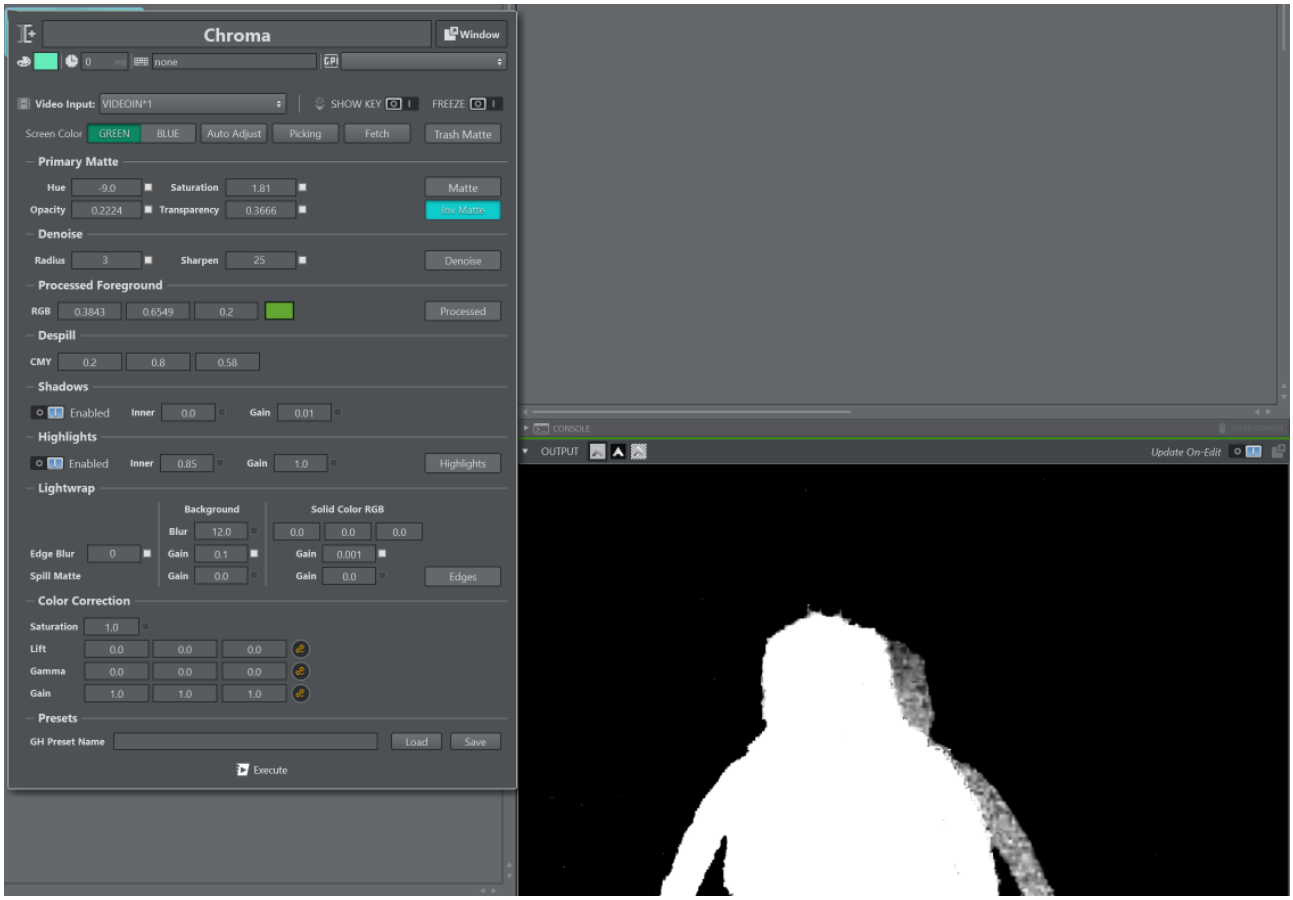
The final composited output after **Auto Adjust** has been pressed is shown above.

Copyright © 2025 Vizrt



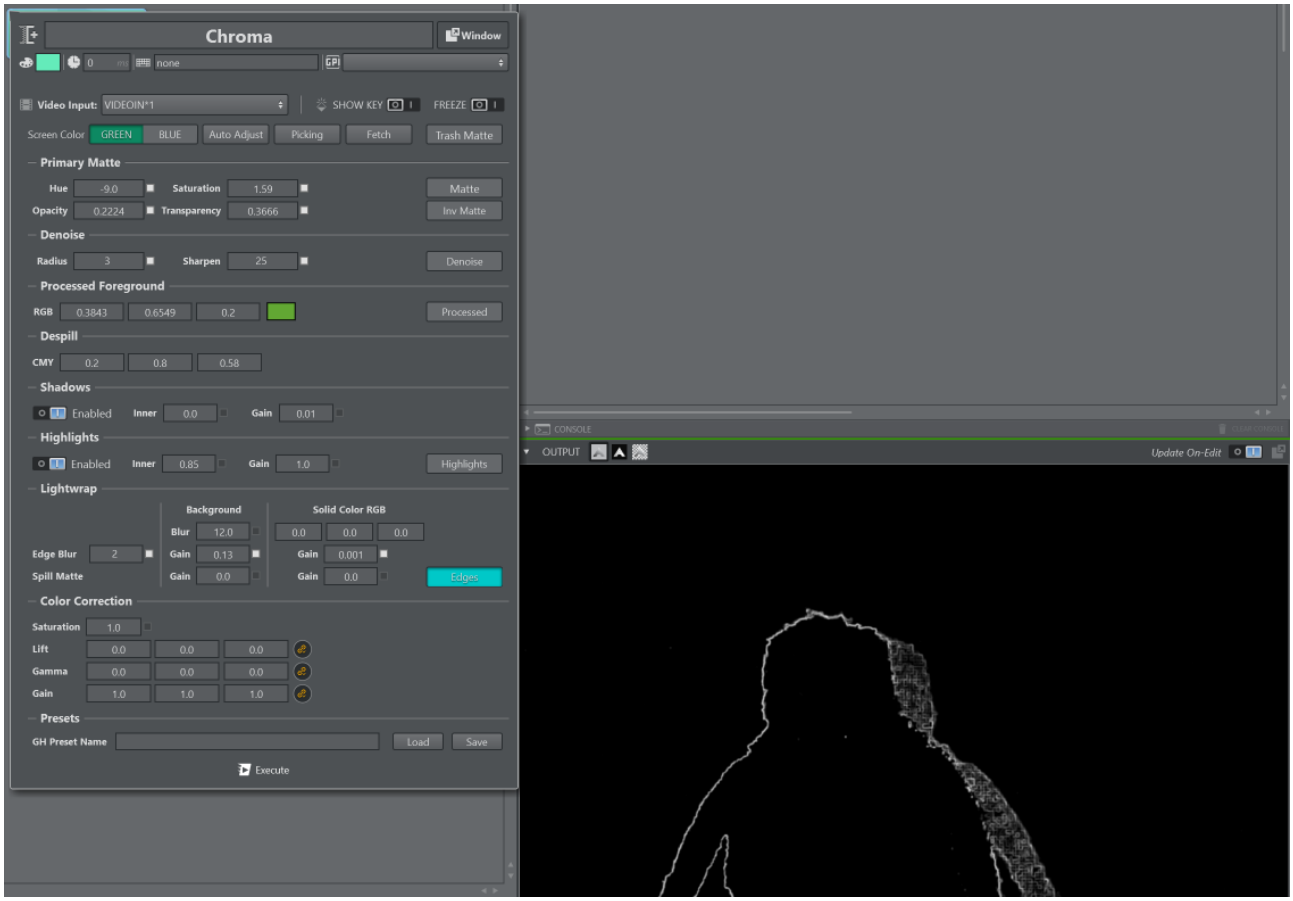
Output of the matte only, black representing foreground and white background, is shown above.

Copyright © 2025 Vizrt



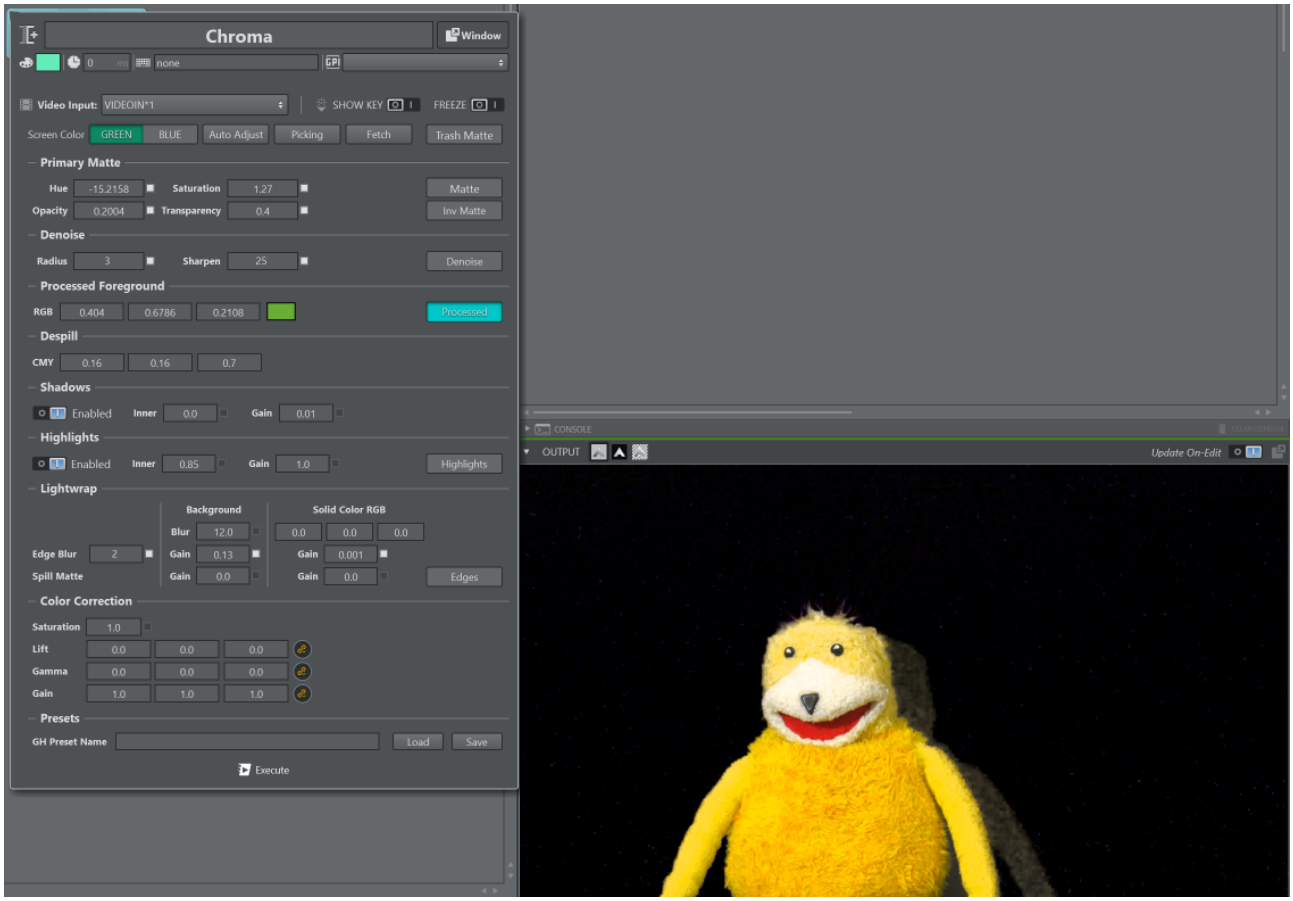
Output of the inverse matte only, black representing background and white the foreground, is shown above; helpful for seeing details such as hair.

Copyright © 2025 Vizrt



Output showing the detected and blurred edge is shown above.

Copyright © 2025 Vizrt

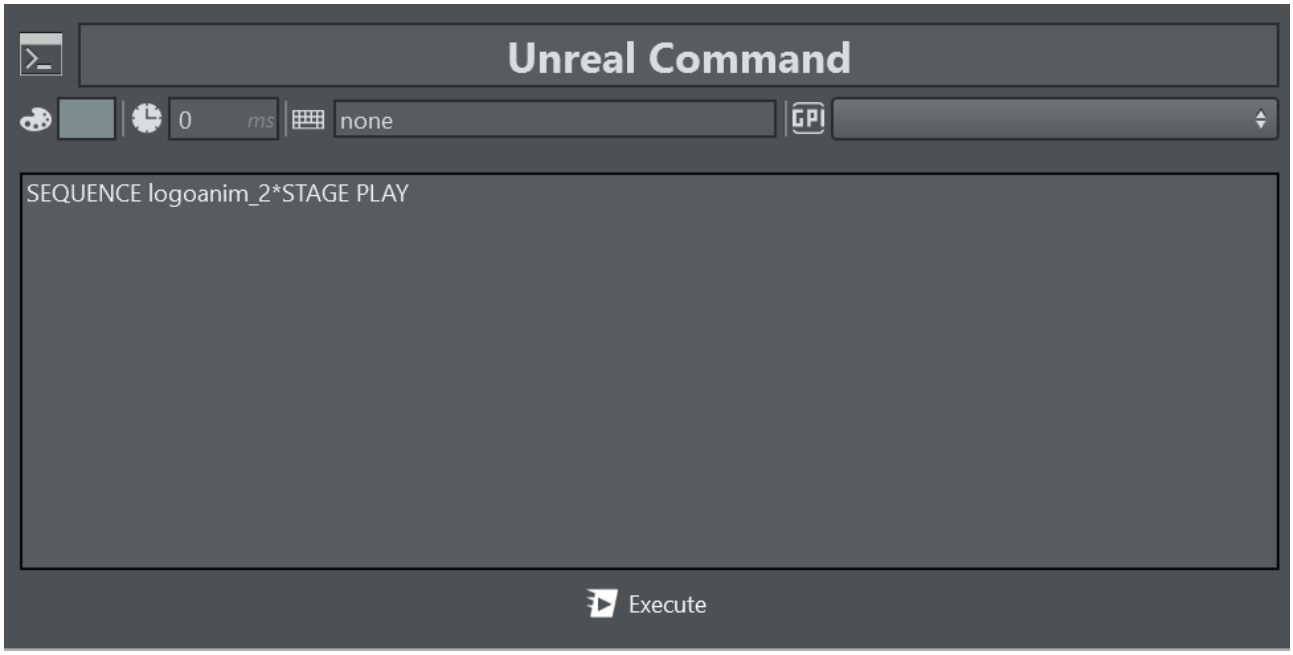


Output of the processed foreground that also shows the light wrap around the talent is shown above.

Unreal Command

The Unreal Command action lets you write custom commands to talk to the Unreal Engine, for example to trigger a sequence. Multiple commands can be sent separating them by **;** and a newline.

Copyright © 2025 Vizrt

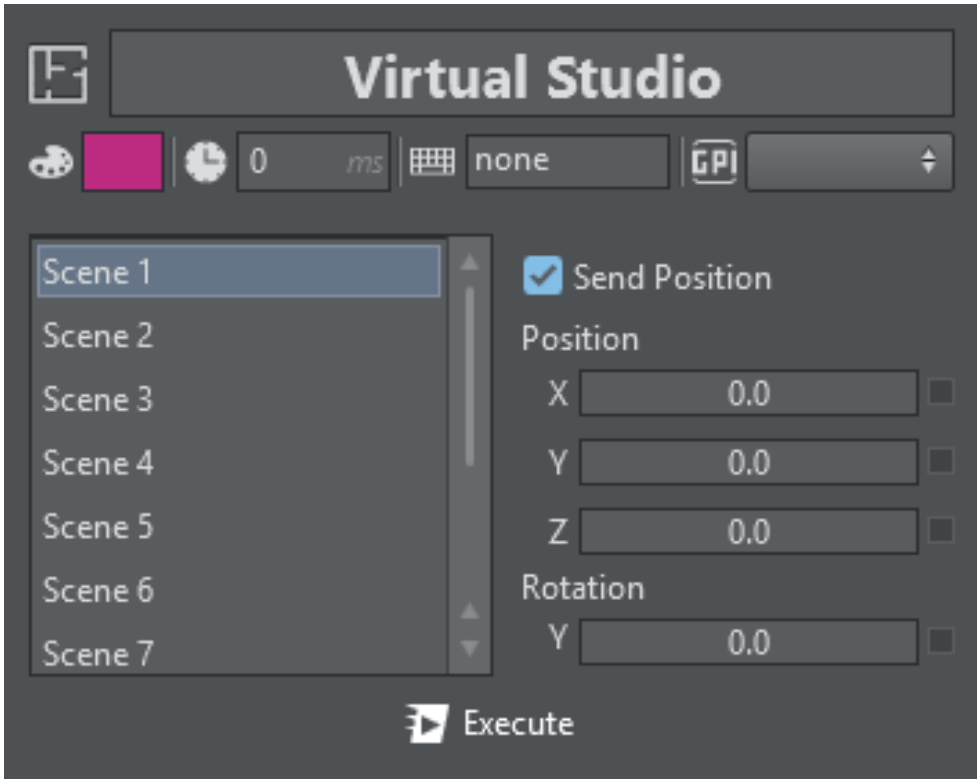


Native Unreal commands are also allowed such as for example:

- **disableallscreenmessages:** Disables all screen messages.
- **enableallscreenmessages:** Enables all screen messages.
- **scalability 2:** Sets the render quality to *High*.
- **stat FPS:** Shows some FPS information on screen.
- **stat unit:** Shows some useful performance information on screen.
- **r.ScreenPercentage 50:** Sets the Screen Percentage scaling resolution to 50.
- **sg.ShadowQuality 0:** Reduces the Shadow Quality to 0.

Copyright © 2025 Vizrt

Virtual Studio



Virtual Studio lets you set up X/Y/Z positions and Y rotations of every scene listed at the left of the panel. When **Send Position** is unchecked, it only sends the scene index to be selected on the Engine(s) and uses the position and rotation stored in the current scene.

Copyright © 2025 Vizrt

Viz Camera



In Viz Camera, you can select between every camera in Viz Engine (from 1 – main to 16). The default behavior once a Camera Action is created is to switch the active camera.

Copyright © 2025 Vizrt



If the toggle **Editor/Remote** is activated you can force a camera to be in **Remote** or in **Editor** mode, for example.

Copyright © 2025 Vizrt



When **Editor** mode is selected, you can enable all aspects of the camera such as **Angle**, **Position** and **Direction** and force the displayed values.

Copyright © 2025 Vizrt

5.4.2 Scene Action Types

Import from a scene/stage tree and edit parameters of these supported types:

- Alpha
- Control Channel
- Control Objects
- ControlObjects in Unreal Engine
- Director
- Geometry
- To Import a Geometry
- Light
- Material
- Image
- Key
- Omo
- PBR Material
- Phong
- Script
- Text
- Transformation
- To Create a Linked Action
- Viewport
- Visibility

Alpha

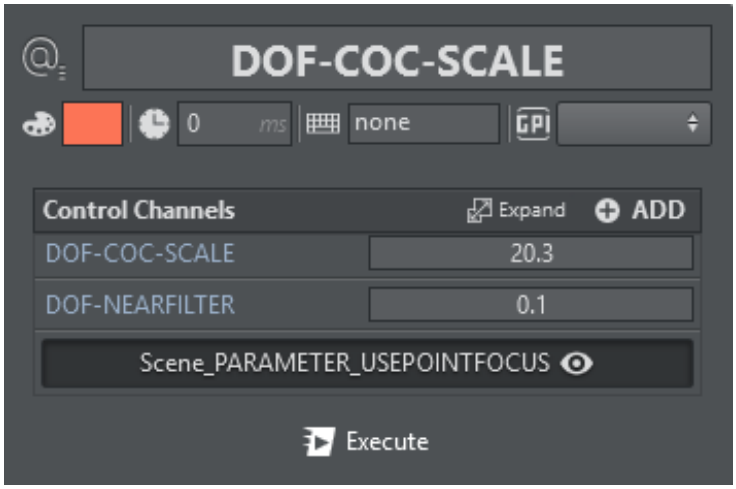
Alpha lets you set the opacity to the referred object, from `0.0` (not visible) to `100.0` (fully visible).



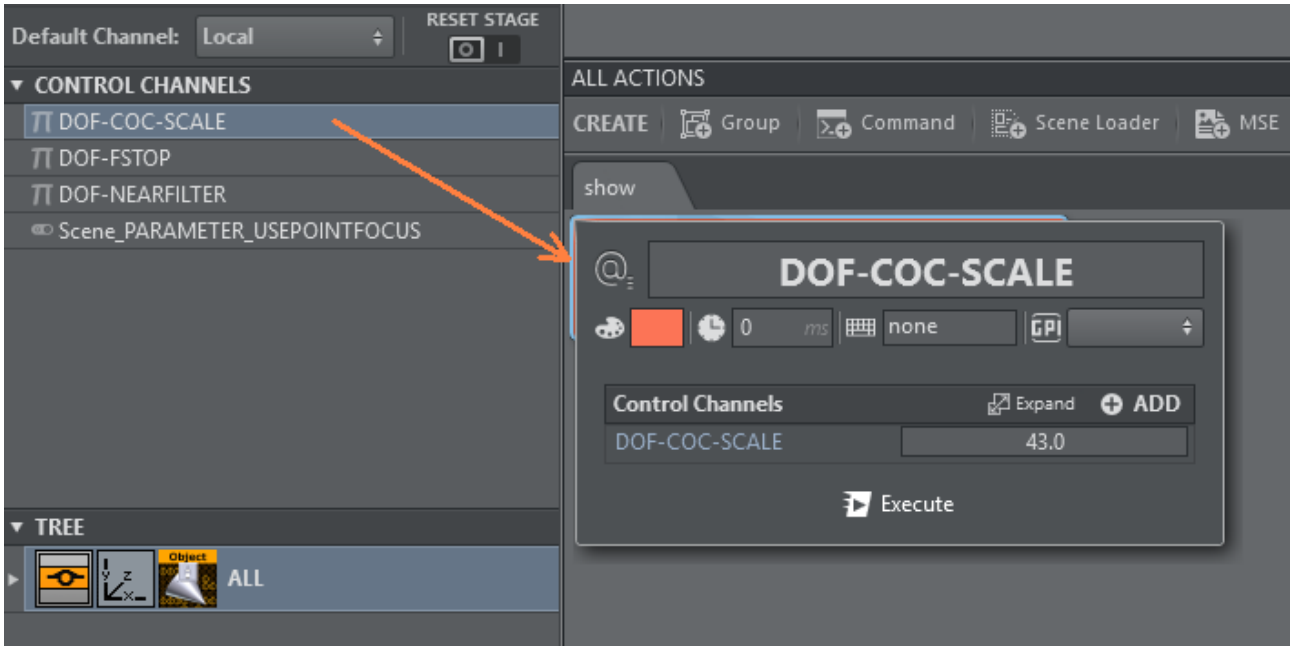
Control Channel

Control Channel lets you control basic Control Channel types such as floats, integers, and Boolean values. This can be particularly handy in conjunction with the Viz Engine 4 Render Graph.

Copyright © 2025 Vizrt

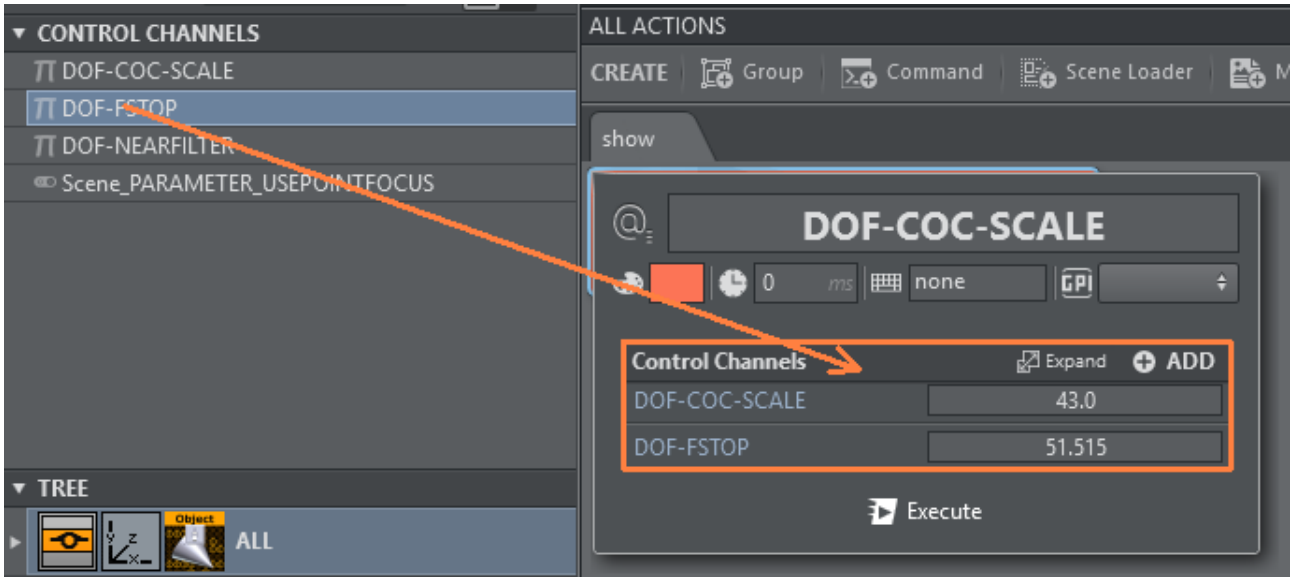


To create a Control Channel action, drag and drop a Control Channel to the action canvas:



If more than one parameter needs to be controlled by the same action, edit the action and drag more parameters into the designated area:

Copyright © 2025 Vizrt

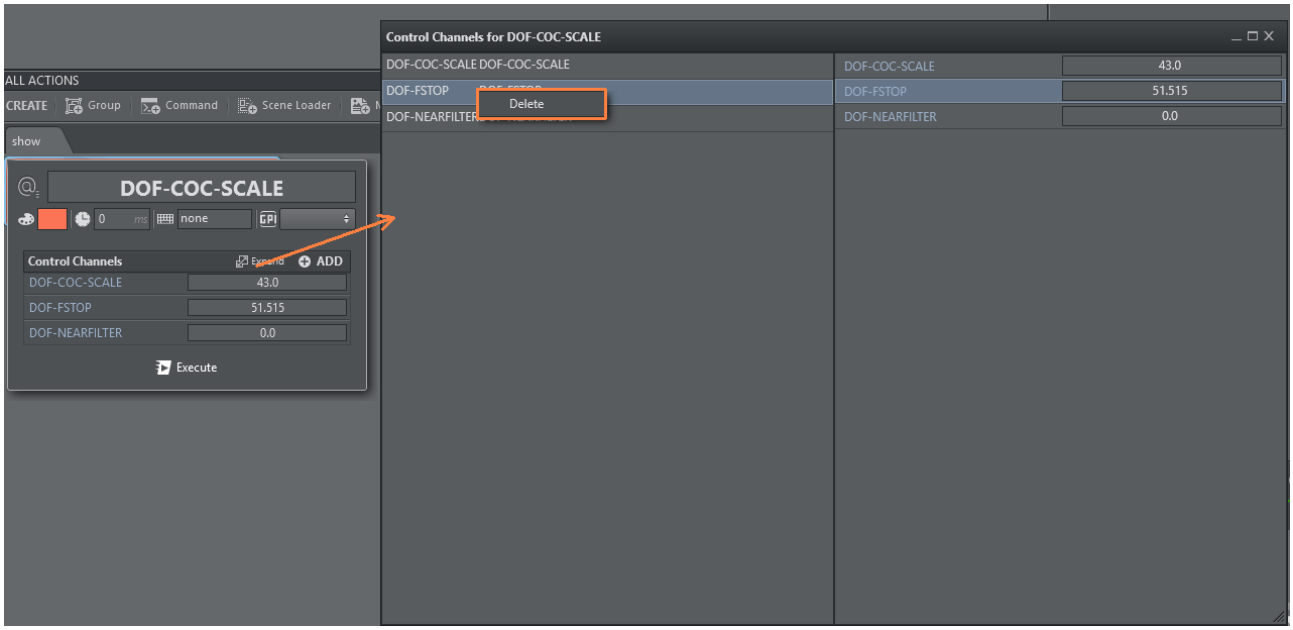


or use the **ADD** button to select from a list of available Control Channels to be added:



To remove Control Channels from the action, click the **Expand** button and use the context menu to delete single Control Channels in the newly opened window:

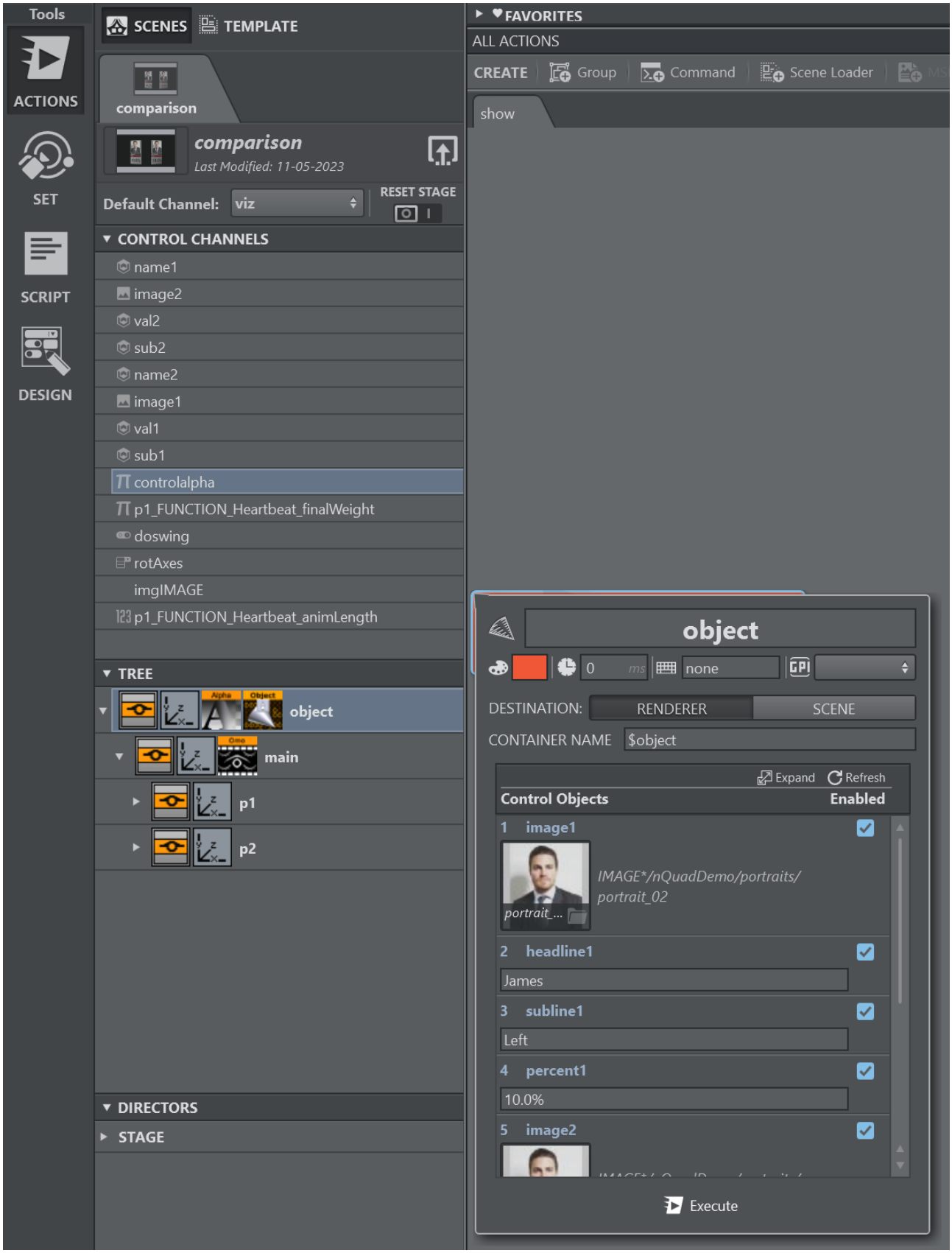
Copyright © 2025 Vizrt



Control Objects

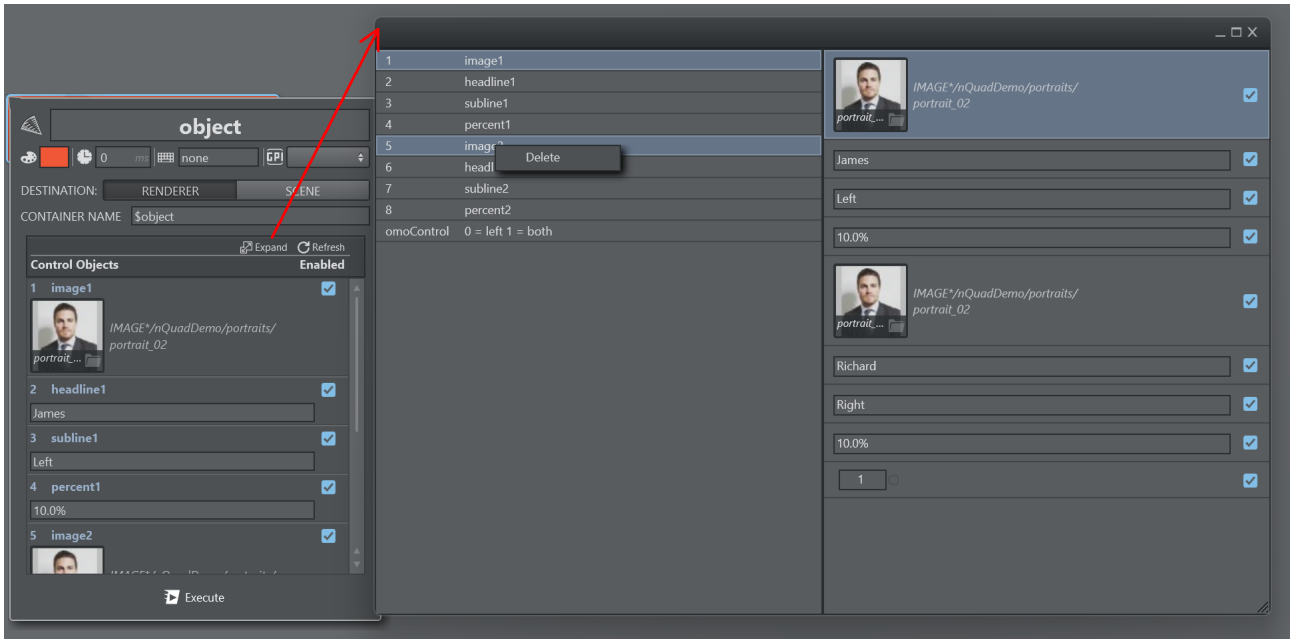
Control Object actions can be used by dragging a Control Object plug-in from the scene tree into the action canvas. It gives you a list of the Control Objects under the respective tree structure.

Copyright © 2025 Vizrt



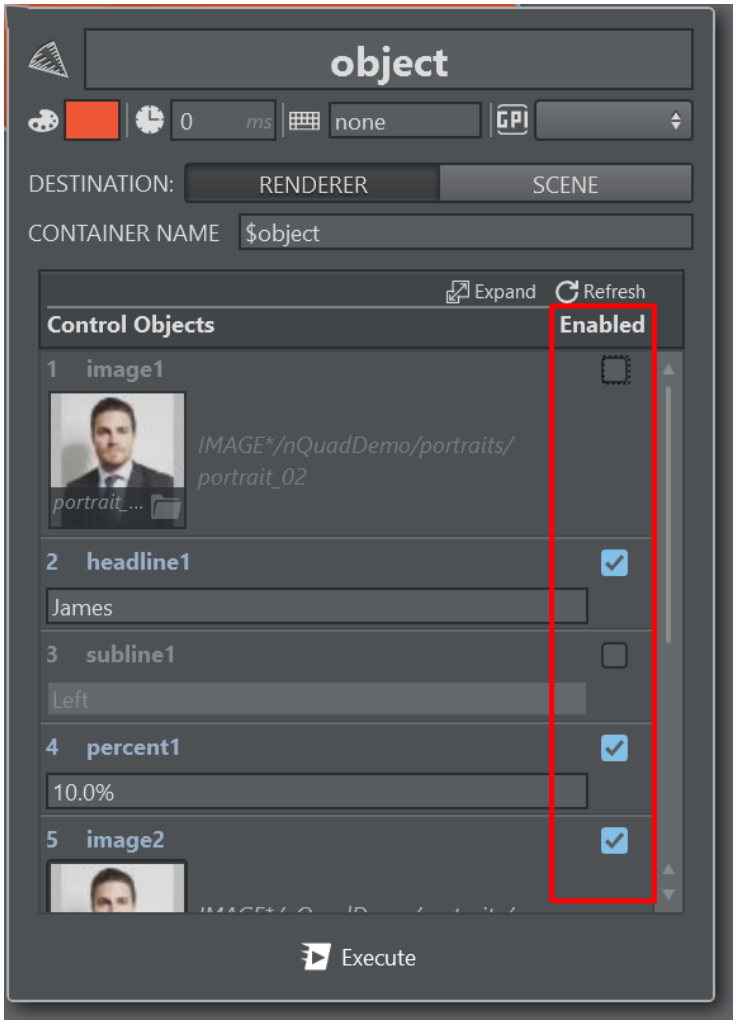
Copyright © 2025 Vizrt

Expand the controls clicking the **Expand** button for a better overview and to enable deletion of single or multiple Control Objects. Select all the controls that shall not be set by this action, then select **Delete** from the context menu.



Use the **Refresh** button to update the list from the associated scene, in case the scene has been modified after the action's creation.

Copyright © 2025 Vizrt

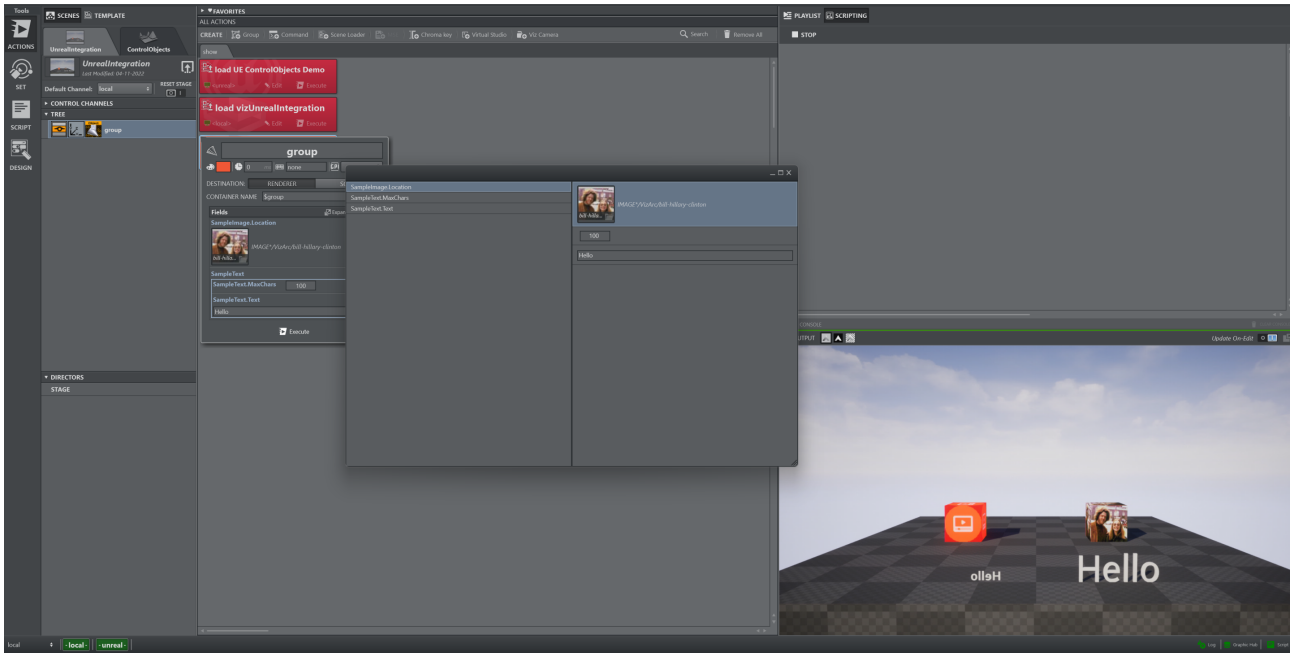


In some cases you might want to send only a subset of the entire Payload. Use the **Enabled** toggle flag for each Control Object to enable or disable setting the specific property on the Viz Engine. It behaves similar to the delete option described above, but in this case you can re-enable the disabled Control Object easily.

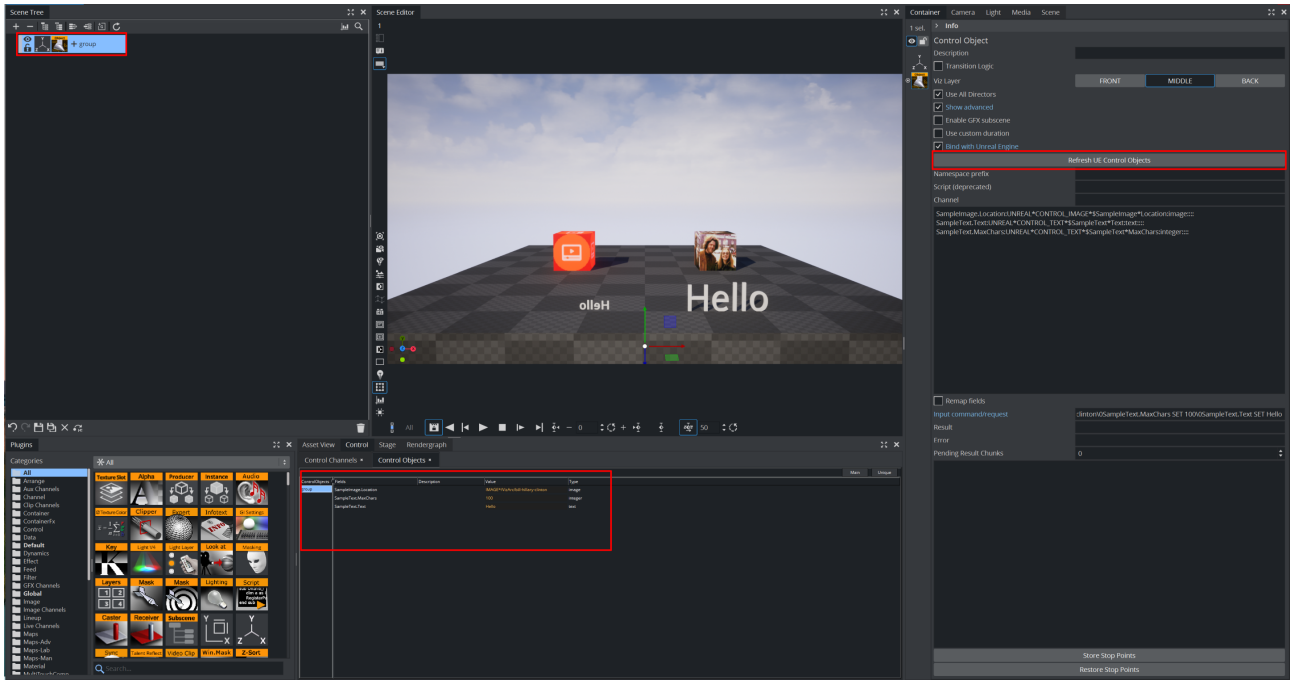
ControlObjects in Unreal Engine

The Viz Engine's Control Object integration allows you to control aspects of the Unreal scene using **Control Objects.** Viz Engine's Control Objects can be used to ingest images from the Graphic Hub to the Unreal Engine.

Copyright © 2025 Vizrt

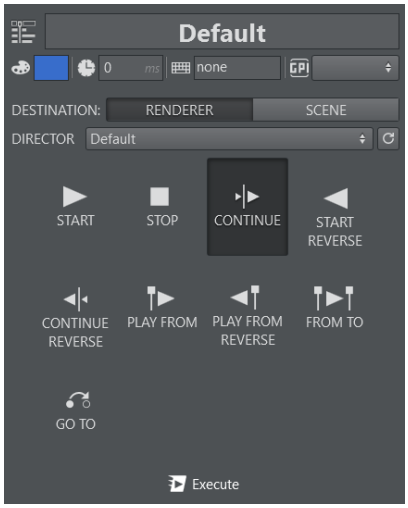


Control Objects can be refreshed within the Viz Engine by using the **Refresh UE Control Objects** fetching the exposed controls of the currently loaded Unreal Level.



Director

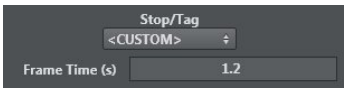
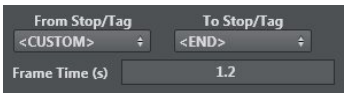
Director lets you handle stage animation of a specific stage director or ALL stages.



It's possible to:

	**START** an animation.
	**STOP** an animated director.
	**CONTINUE** an animation to the next step.
	**START** in **REVERSE** mode.
	**CONTINUE** in **REVERSE** mode.
	**PLAY FROM** a specific KEYFRAME.
	**PLAY FROM** a specific keyframe in **REVERSE** mode.
	**PLAY** an animation **FROM** a specific KEYFRAME **TO** another.

Copyright © 2025 Vizrt

	**SET** an absolute time in seconds and jump (**GO TO**) tag/stop points.
	This menu lets you set the **Stop/Tag** for **PLAY FROM**, **PLAY FROM REVERSE** and **GO TO** options, and the **Frame Time** for the **CUSTOM** option.
	This menu lets you set the **From stop/Tag** and **To Stop/Tag** for playout of an animation for the **FROM TO** option.

The Director Action shows the number of stop points when a specific director is selected (for example, not the **<STAGE>**). When the action is selected, a progress bar on top alongside with the currently reached stop point (for example, 2/3) is visualized. The number of stop points is read on the creation of the action and when the action is explicitly refreshed.

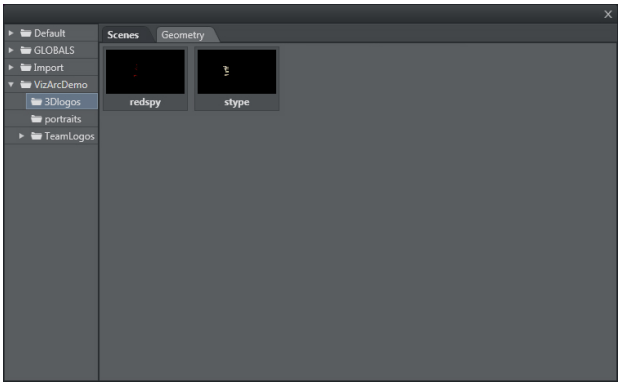


Geometry

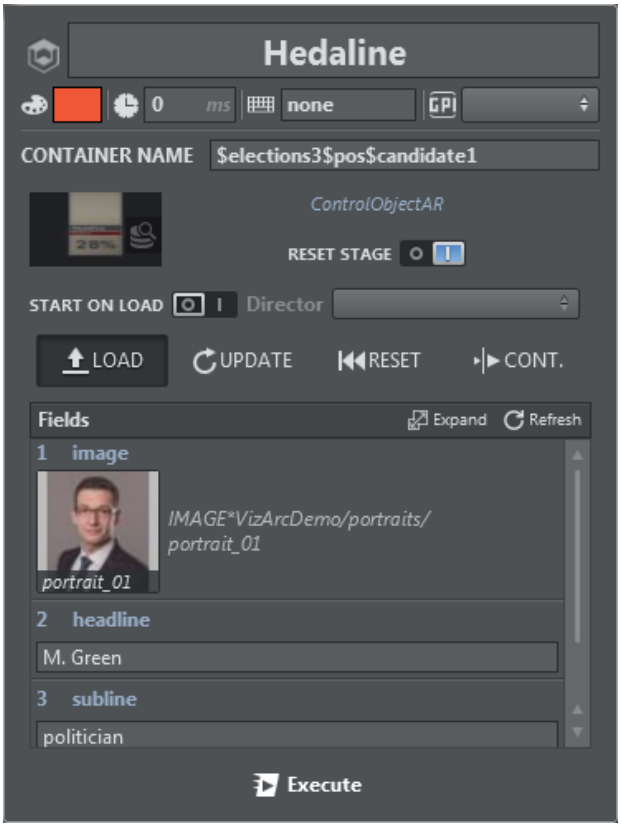
Use this menu to set an OBJ or SCENE element in a specified container.



Copyright © 2025 Vizrt

	Clicking **Select Geometry or Scene** makes all GH trees appear in a new window, where you can select a new GEOM.
	Selecting **OBJECT** makes the relevant thumbnail appear in the action box.

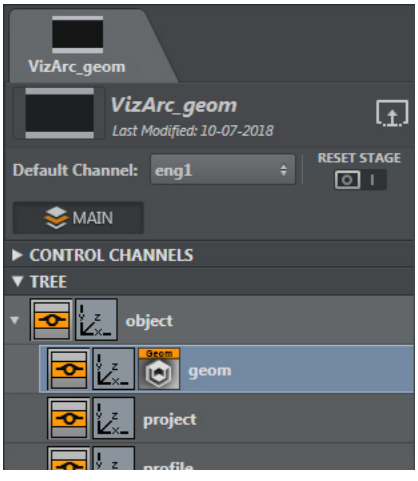
Copyright © 2025 Vizrt



It's also possible to select a **SCENE** instead of GEOM. Once the scene is loaded in the geom plugin, a separate edit section appears where it's possible to handle ***control object*** (if applicable) of the selected scene.

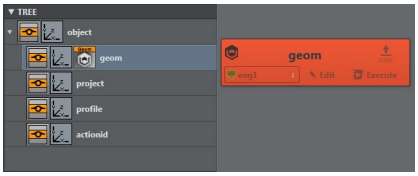
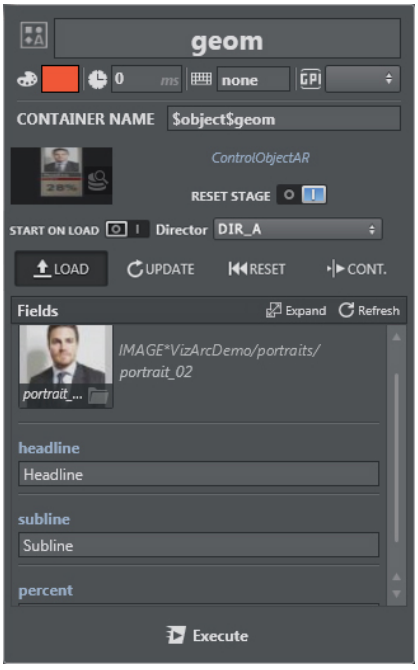
To Import a Geometry

In Viz Arc, it's possible to import objectified scenes (GEOM) into another scene through the GEOMETRY element. Once the selected object is imported in the right GEOM, it's possible to load, update, refresh and continue. A step-by-step example of importing and playing an objectified scene follows below:

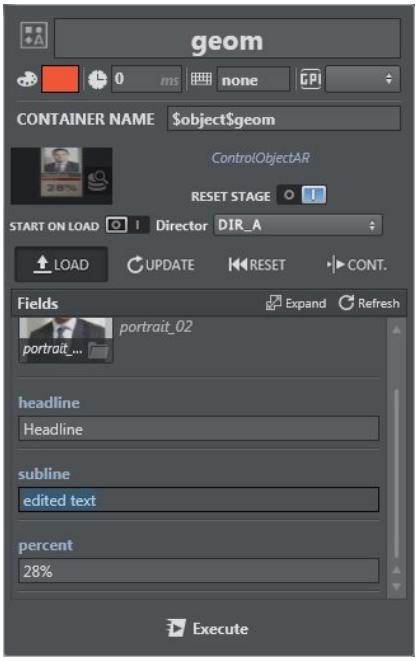


Load a main scene that contains at least one GEOM element.

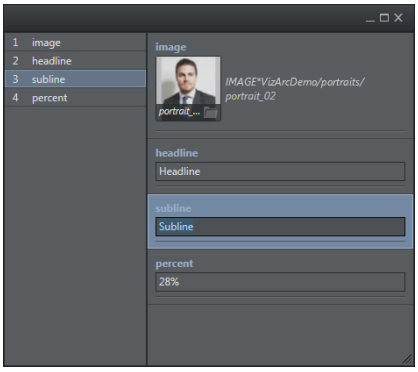
Copyright © 2025 Vizrt

	Drag the GEOM element into the main area.
	Press **Edit** and select the scene.
	Once you select the scene, a list of control objects appears (if applicable). You can enable the **RESET STAGE** option to have the scene placed on the 0.0 frame in the stage.

Copyright © 2025 Vizrt

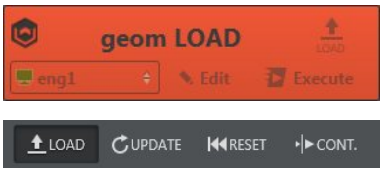


You can edit every control object listed in the scene.



You can use the Expand icon to open a popup window that contains a detailed list of control objects, where you can modify each field by clicking it.

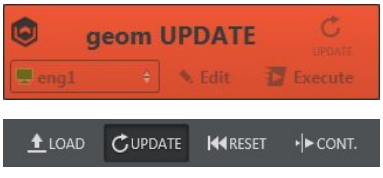
Once you have created your action box, you can duplicate it by pressing **CTRL +D** and assign a different action (**LOAD**, **UPDATE**, **RESET**, **CONT**) to each box. Follow the steps below to play the GEOM scene in the main scene:



This command loads the scene/object in geom.

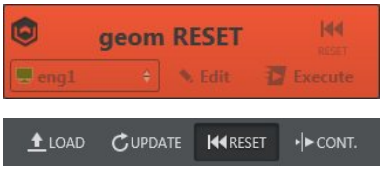
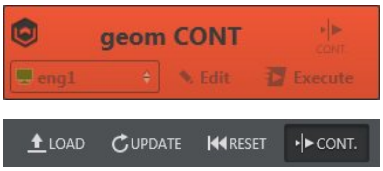
If RESET STAGE is enabled, the scene/object is loaded with all its stage animation set to frame 0.

If RESET STAGE is disabled, it works like a *Take command*, loads and start stage.

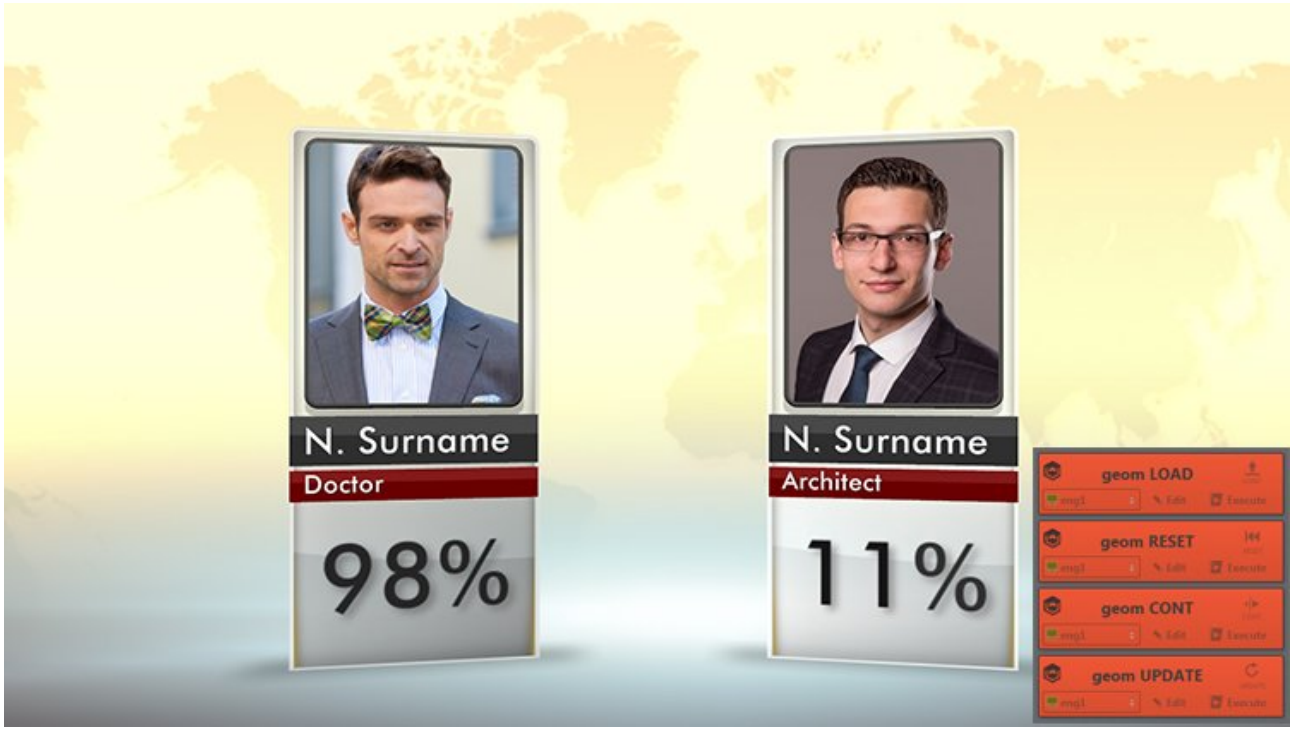


This command updates all changed control objects On Air.

Copyright © 2025 Vizrt

	This command initializes the stage (*frame 0.0*).
	This command plays the scene. If the scene has multiple animation steps, you can continue the animation by clicking it multiple times.

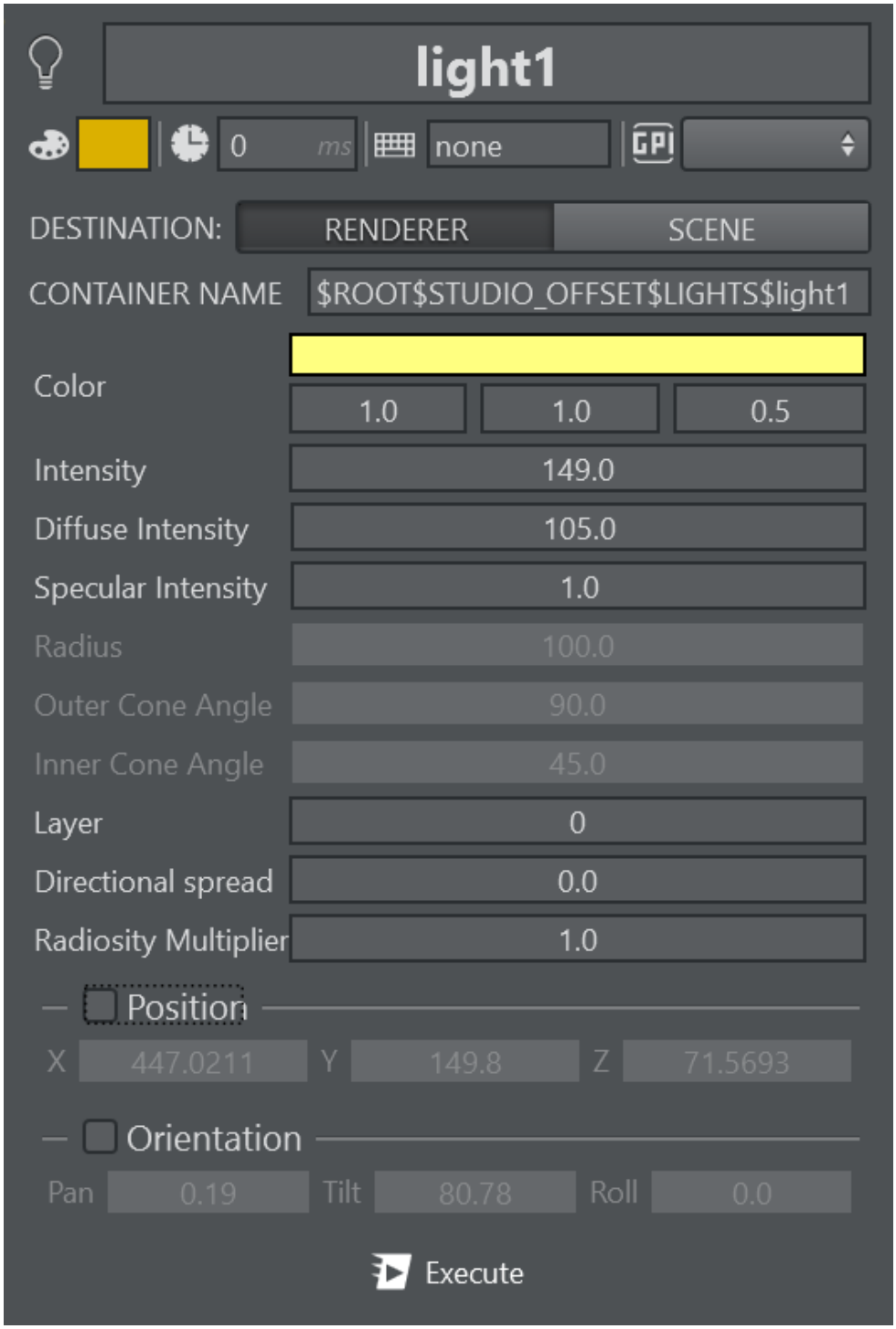
For **LOAD**, **RESET** and **CONTINUE**, you can choose a specific director instead of the full stage. 



Light

The light action works together with Viz Engine V4 lights. All light types are supported (directional, area, spot, point).

Copyright © 2025 Vizrt

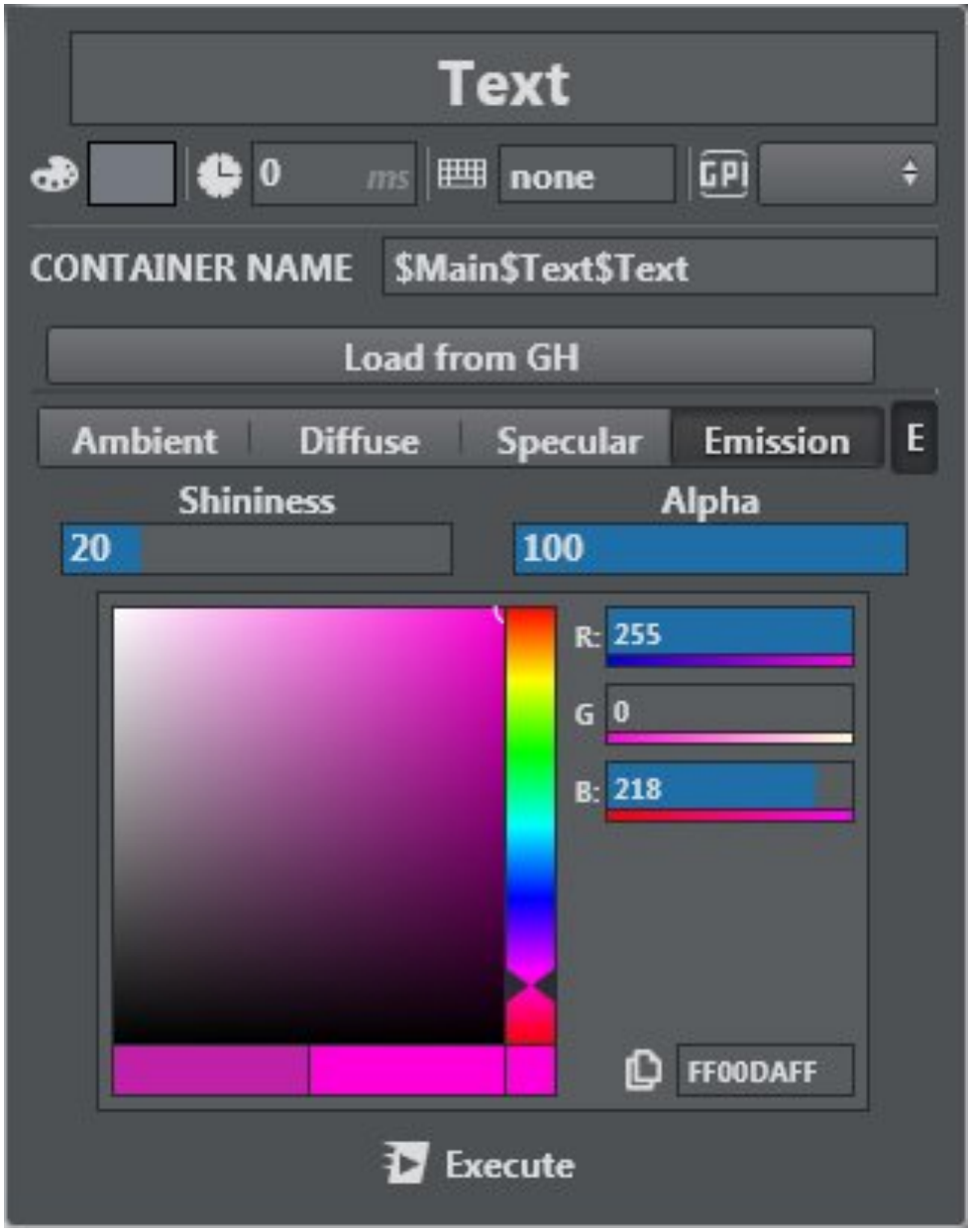


The **Position** and **Orientation** settings can be optionally set and act on the container transformation.

Copyright © 2025 Vizrt

Material

The Material menu lets you set the material of an element and its **Shininess** and **Alpha**. Set a unique color or set every color to **Ambient**, **Diffuse**, **Specular**, **Emission** lighting by enabling/disabling the **E** button next to the lighting tabs.



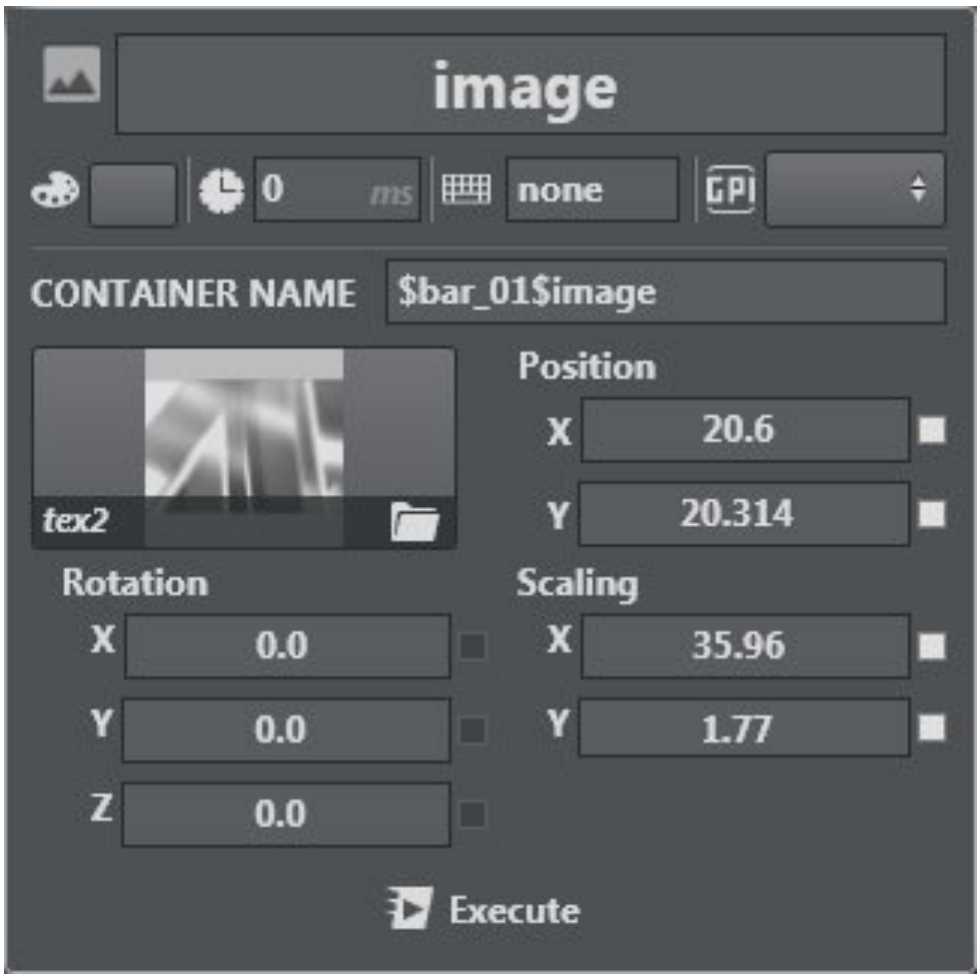
- **Ambient:** Determines the color that the object reflects when illuminated by a natural ambient light.
- **Diffuse:** Determines the real color of the object, revealed when the object is illuminated by a neutral light.
- **Specular:** Determines the color that the object has when the light meets the object surface and reflects in the camera.
- **Emissive:** Determines the self-illumination color.

You can import an existing material in GH (Graphic Hub) by pressing the **Load from GH** button.

Copyright © 2025 Vizrt

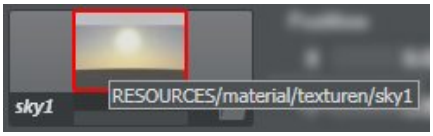
Image

Image lets you select an image, both from GH tree and external explorer files, and set basic transformation parameters like **Position**, **Rotation**, and **Scaling** of the image texture.



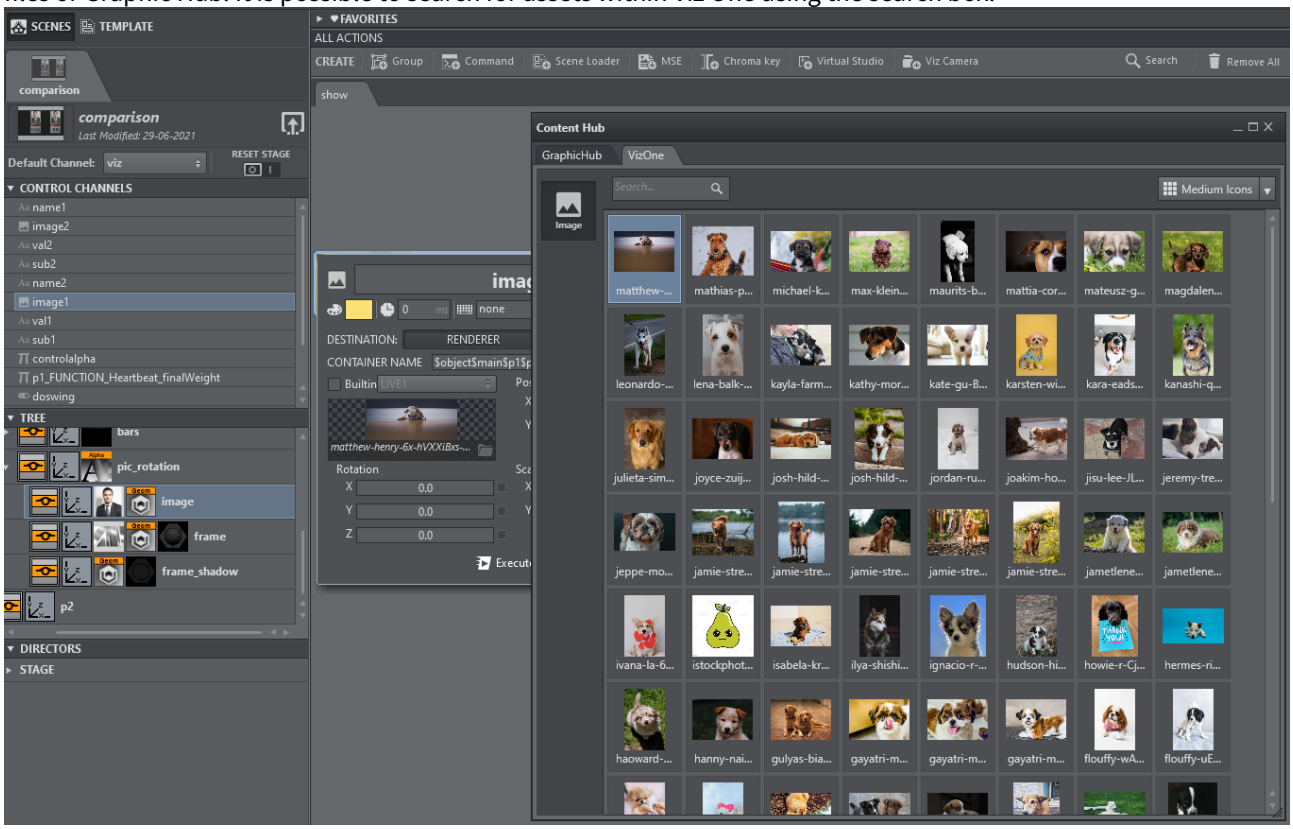
	Once in edit mode, click **image** to browse the database and select an image from Graphic Hub.
	Once in edit mode, click the folder icon at the bottom of the image box to browse the file system and select an image from your PC or device.

Copyright © 2025 Vizrt



To see the image path, hover your cursor over it.

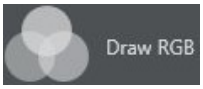
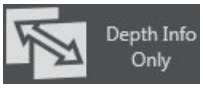
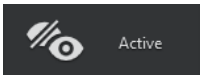
With a properly configured Viz One or Viz Media Service, assets can be picked from those sources in addition to local files or Graphic Hub. It is possible to search for assets within Viz One using the search box.



Key



Use the Key menu to handle Key options for a container:

	**Draw RGB**
	**Draw Key**
	**Combine with BG Chroma Key** – combines the alpha of the container with the alpha of the background before blending the foreground with the background.
	**Depth Info Only** – depth only for occlusion rendering with Live Video.
	**Active** - shows if the Key plugin is active or inactive. Typically used in a green screen scenario to determine whether an object should appear in front of or behind the talent.

Copyright © 2025 Vizrt

Omo

Use this menu to adjust the Omo plug-in settings:



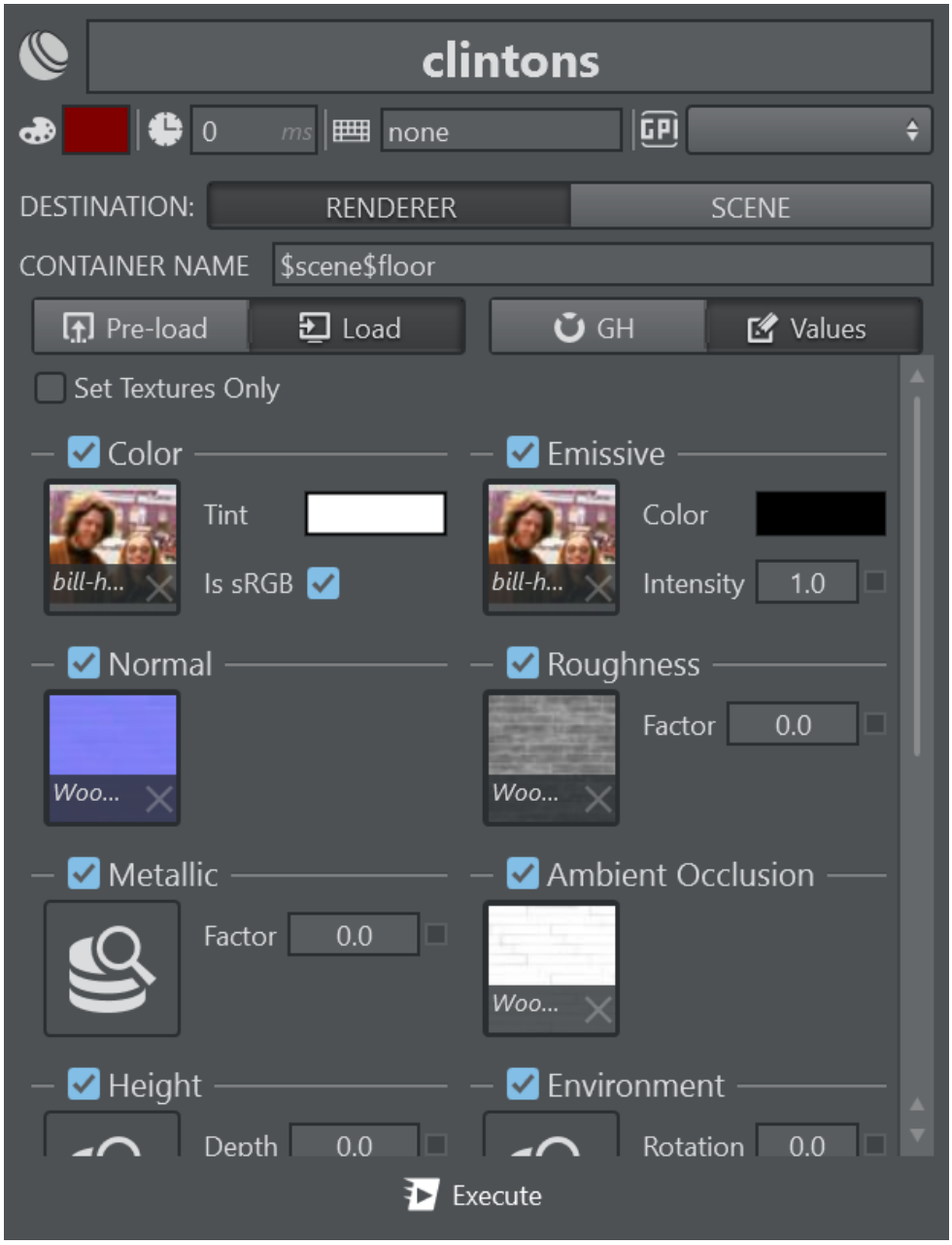
This plug-in displays all sub-containers, either one at a time or by keeping the previous container visible until the selected container appears (by enabling the **Show until** parameter). The **Visible Container** parameter determines which sub-container is displayed.

> **Note:** For more on the Viz Omo Plug-in, see the Viz Artist User Guide.

PBR Material

The PBR Material action works together with Viz Engine V4 materials.

Copyright © 2025 Vizrt

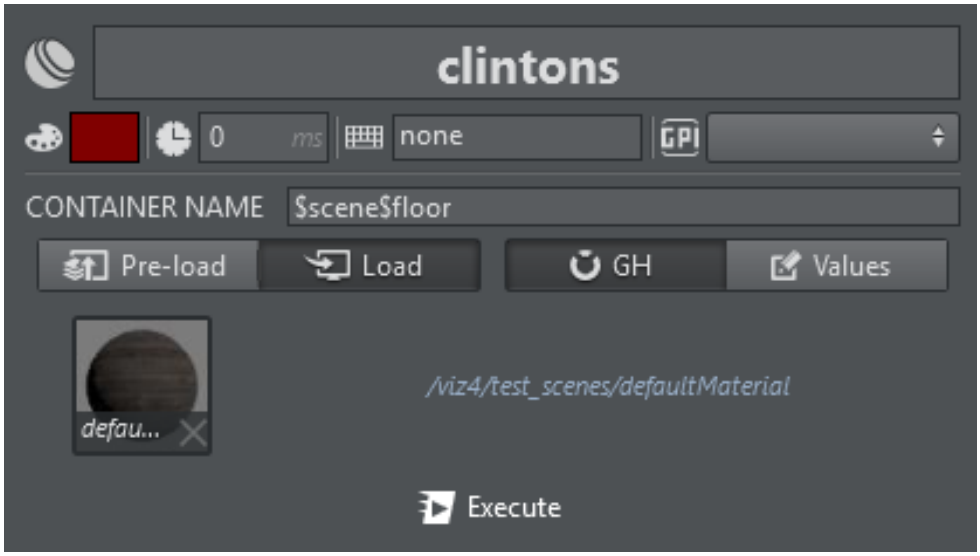


The action offers a subset of Viz Artist's PBR settings, exposing only the most prominent features, such as all textures and colors defining the PBR material.

Enable the **Set Textures Only** checkbox to send texture information only, use the respective checkboxes on the various attributes to send to enable or disable them.

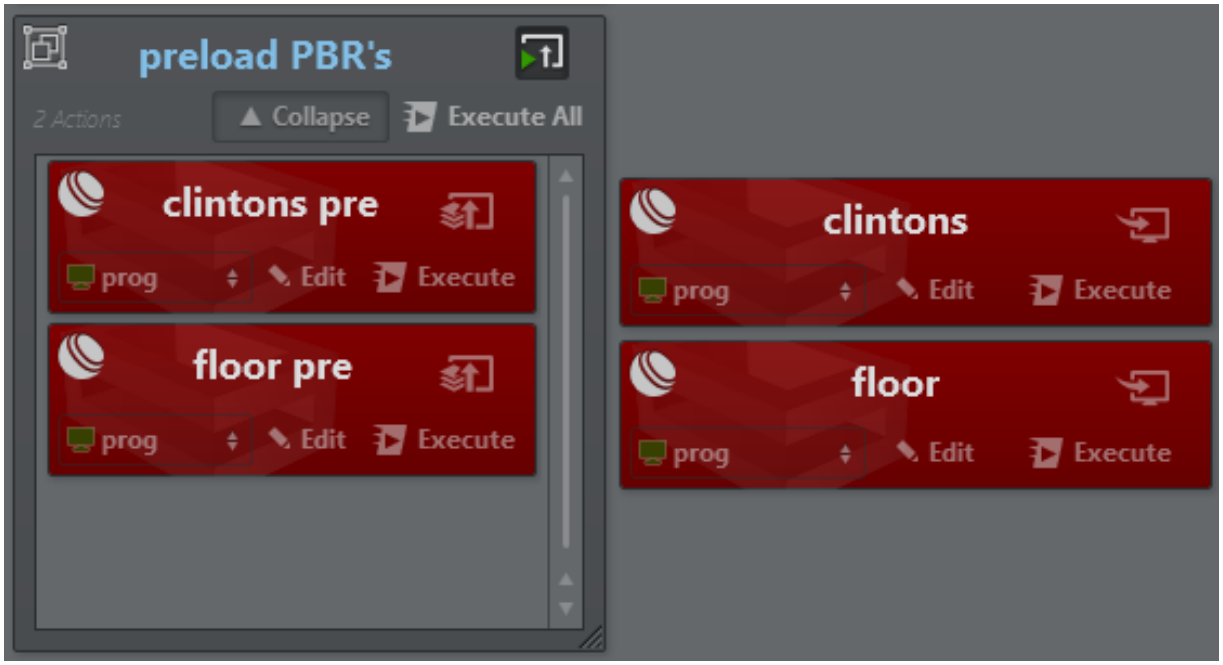
Use drag and drop from any image assets to copy and paste a texture asset to another slot.

Copyright © 2025 Vizrt



When the **GH Mode** is selected, you can browse Graphic Hub for PBR materials and load them through this action.

The action can be executed in two modes: **Pre-Load** or **Load**. **Pre-Load** preloads all the images present in the action on the Engine. **Load** effectively loads the PBR Material onto the Engine. Typically, you group all Pre-Load actions and execute them on Initialization, then trigger only the **Load** actions. Right clicking the **action execute** button always sends **Pre-Load** commands.

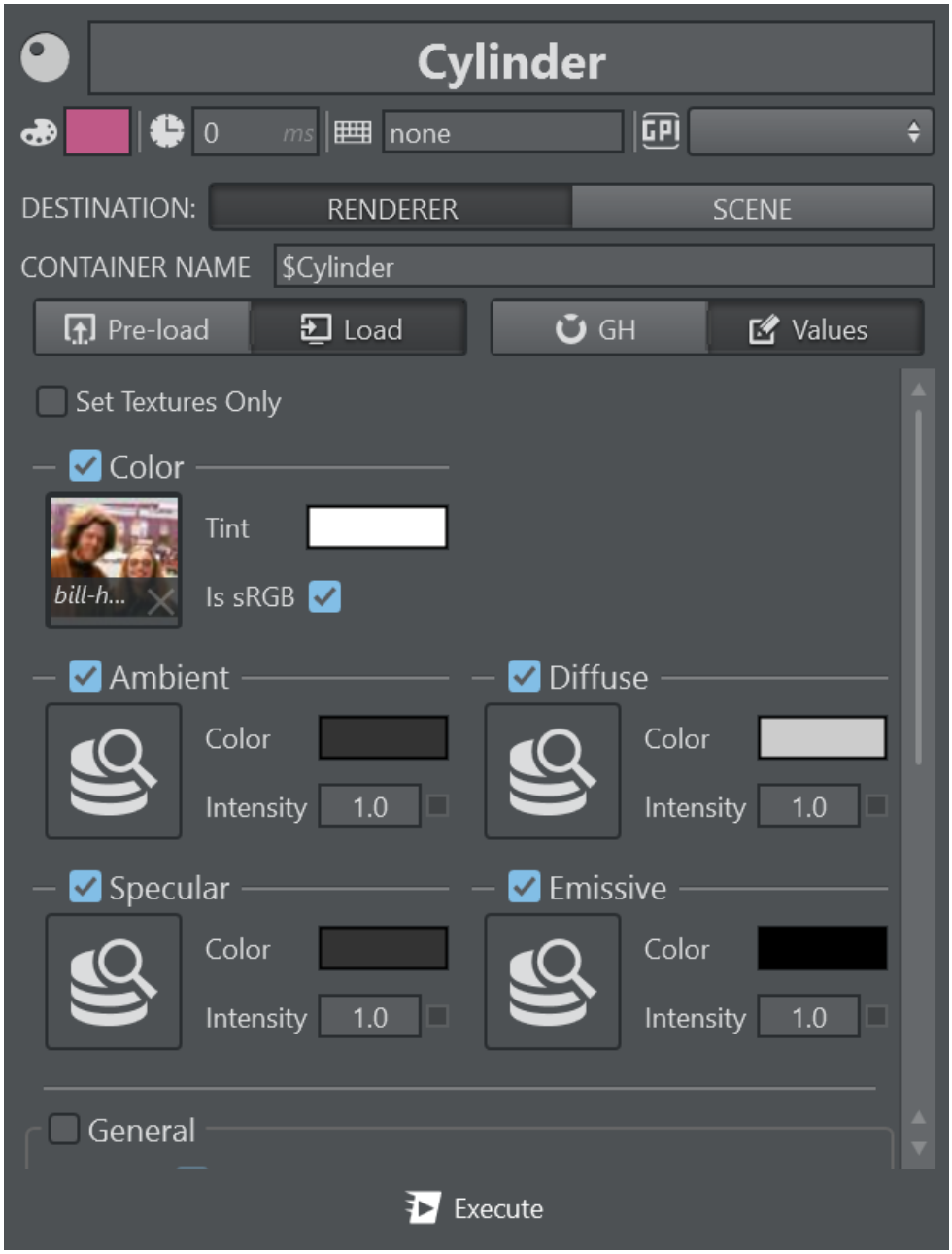


The screenshot above shows two PBR actions in an initialization group as pre-loading and the same actions on the right for loading.

Phong

The Phong Material action works together with Viz Engine 4.3 and newer.

Copyright © 2025 Vizrt



The action offers a subset of Viz Artist's Phong material settings, exposing only the most prominent features, such as all textures and colors defining the Phong material.

When **GH Mode** is selected, you can browse Graphic Hub for Phong materials and load them through this action.

Analog to the PBR Material, the action can be executed in two modes: **Pre-Load** or **Load**. **Pre-Load** preloads all the images present in the action on the Engine. **Load** effectively loads the Phong Material onto Viz Engine. Typically, you group all Pre-Load actions and execute them on Initialization, then trigger only the **Load** actions. Right clicking the **action execute** button always sends **Pre-Load** commands.

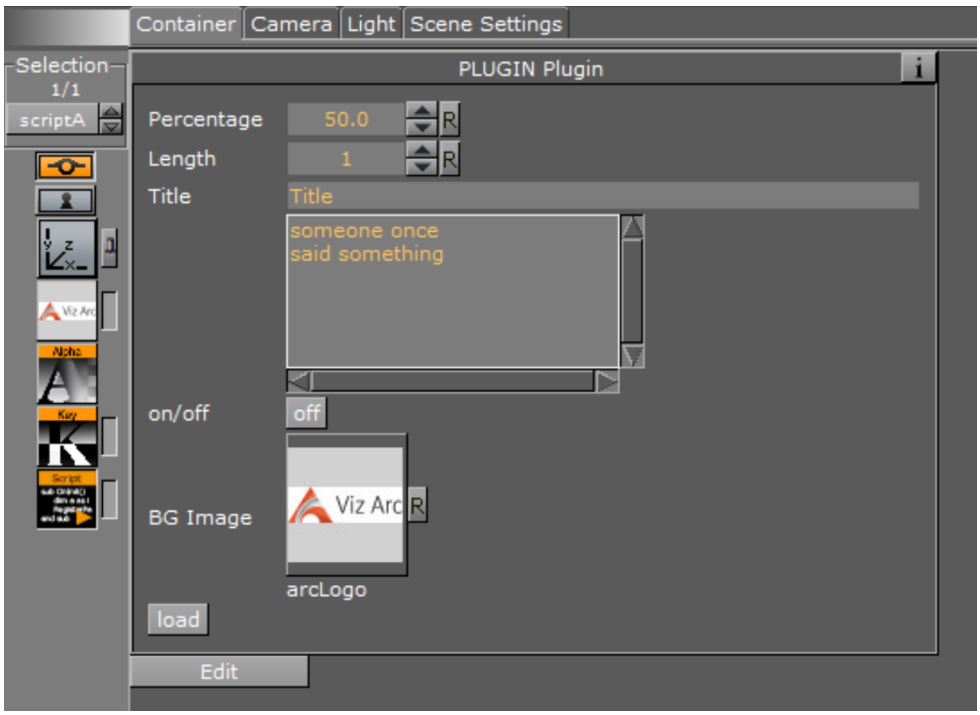
Enable the **Set Textures Only** checkbox to send texture information only, use the respective checkboxes on the various attributes to send to enable or disable them.

Use drag and drop from any image assets to copy and paste a texture asset to another slot.

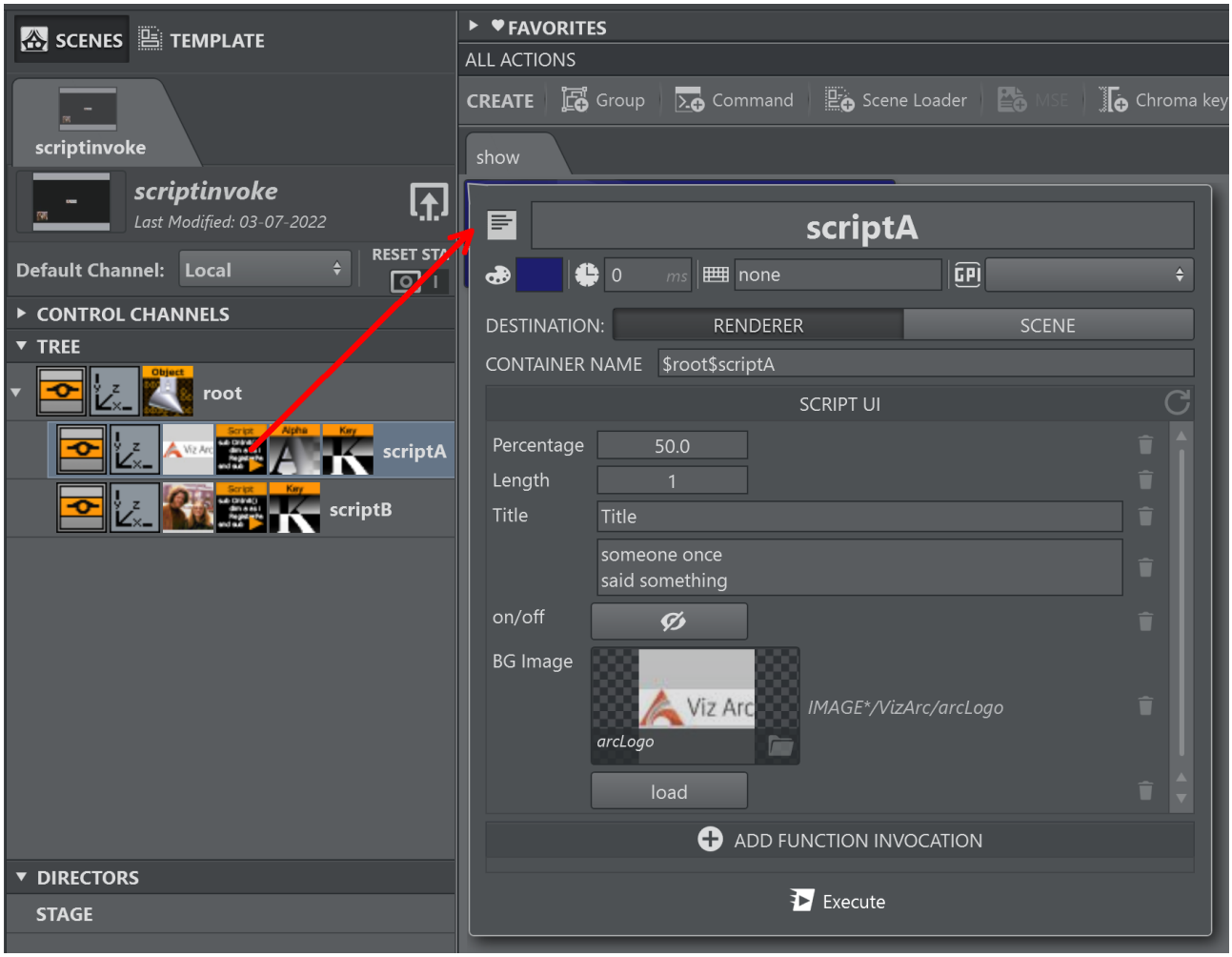
Copyright © 2025 Vizrt

Script

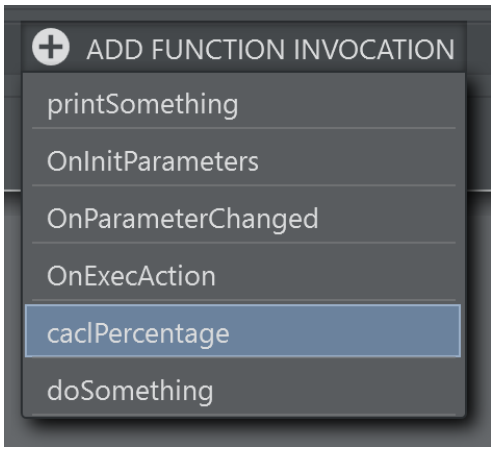
The Script action lets you modify the registered UI parameters of a Viz Engine container script. Additionally, it lets you invoke any of the script's functions with its respective parameters.



An example of a container script UI in Viz Artist.

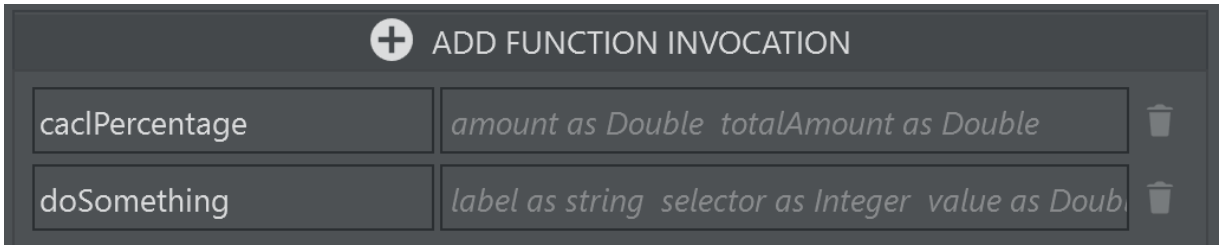


Drag and drop a Script container plug-in from the tree view into the action canvas to create a **Script Action.**

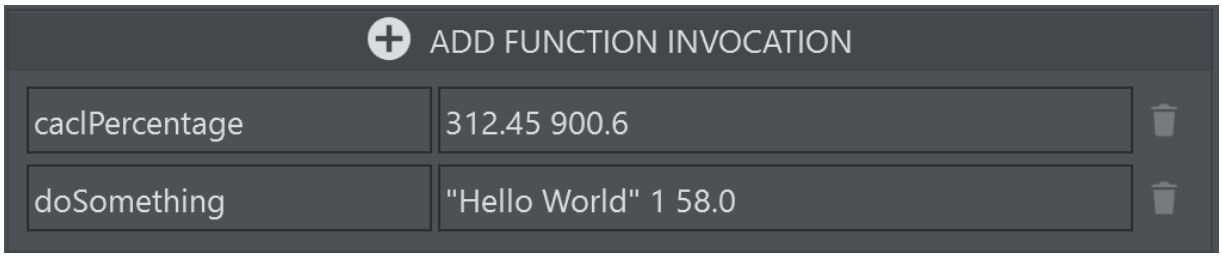


Click on the **ADD FUNCTION INVOCATION** button to add any of the script's functions to the action.

Copyright © 2025 Vizrt

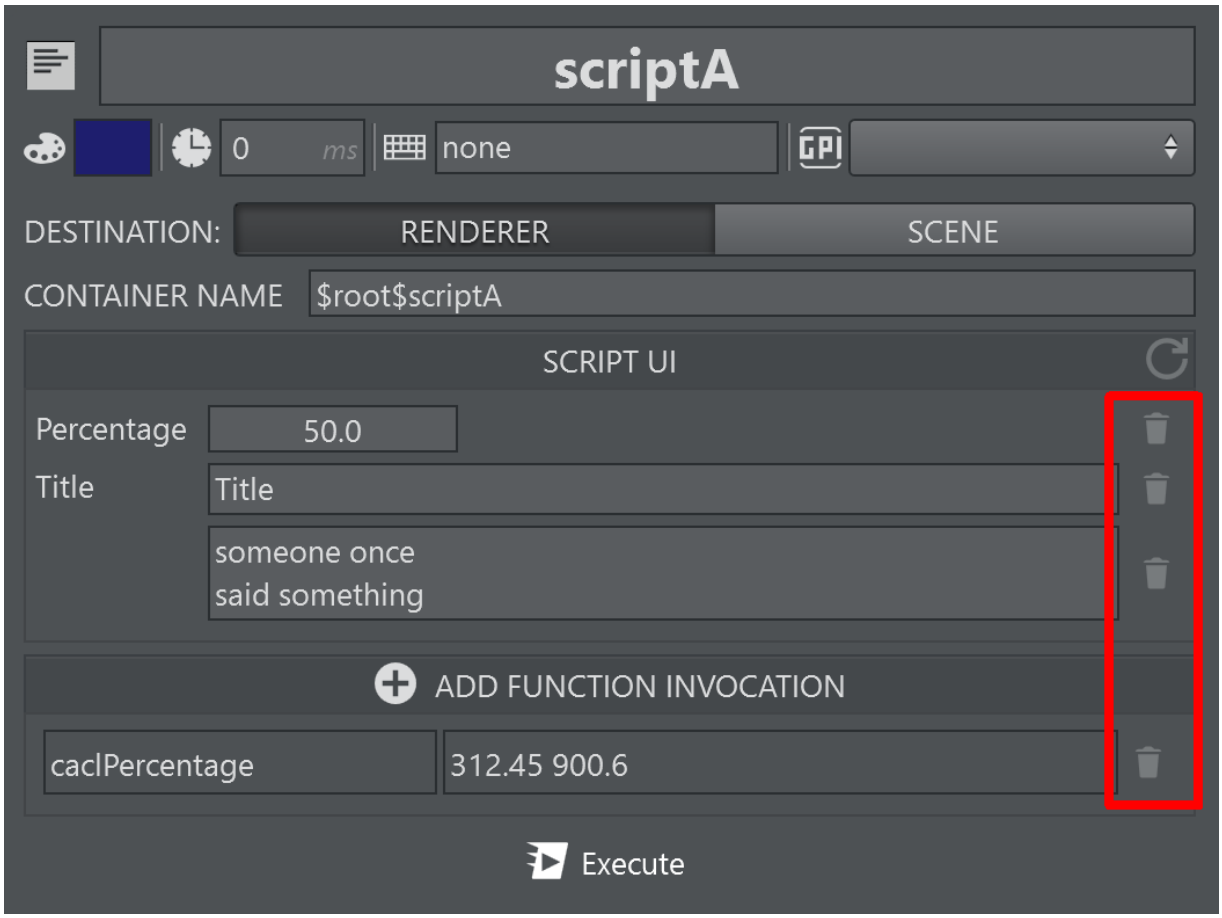


After adding the functions you whish to invoke, enter the parameters where necessary. A hint in the parameter's text box in the parameter gives you the signature of the function.



Note: Parameters need to be separated by a space character. When strings containing spaces are passed as parameters, use the quote **"** character as a delimiter of the string as in the "Hello World" sample above.

On execution of the **Script Action**, all UI parameter values and function invocations listed in the action are sent to the selected channel's Viz Engine(s).

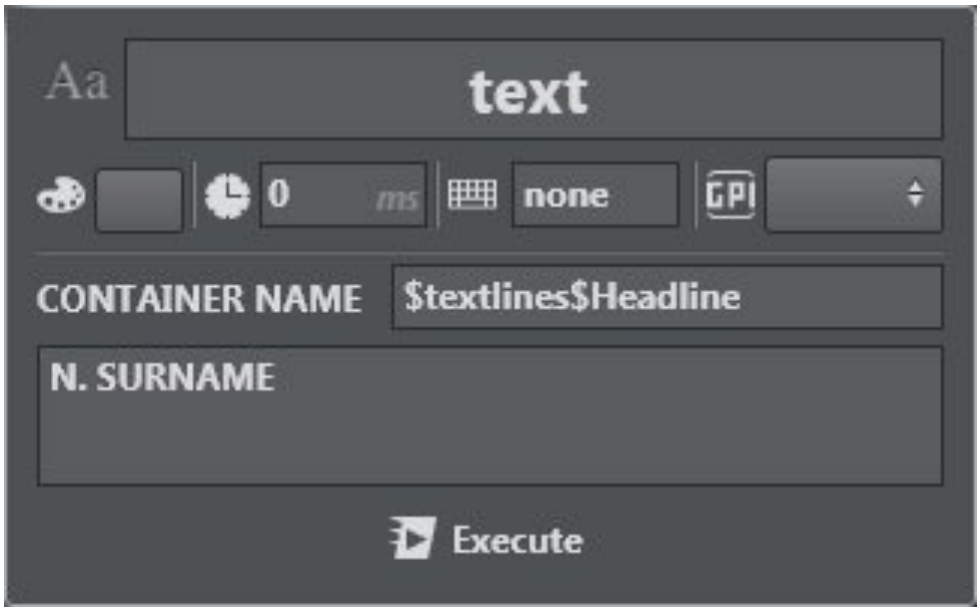


Copyright © 2025 Vizrt

Use the small trash icons on the right hand side of each row to remove parameters that you don't want to send to the Viz Engine. Press the refresh icon to reload the script from the Viz Engine's scene. Make sure the scene is loaded on the **Editing Engine**.

Text

This menu lets you set text in the selected container. The action supports standard text geometry as well as **Text** introduced in Viz Engine 4.3.



Transformation

Transformation lets you set position, rotation and scaling of the selected container. You can also scale in proportion by clicking the **Lock** icon  next to the scaling panel. **Position**, **Rotation** and **Scaling** can be disabled using the check boxes. When disabled, those parts are not being sent to the Engine on execution. For example, when selecting **Rotation** only, only the rotation of the target container is affected on execution.

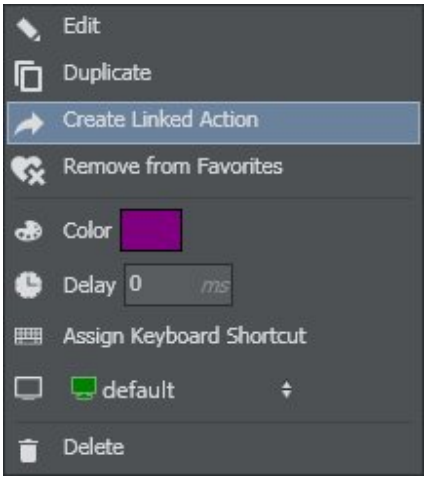
Copyright © 2025 Vizrt

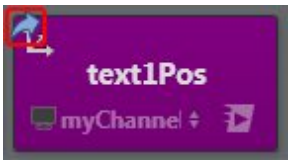


It's possible to create a linked action for TRANSFORMATION actions. This option is useful for controlling different scenes with the same tree structure, for example.

To Create a Linked Action

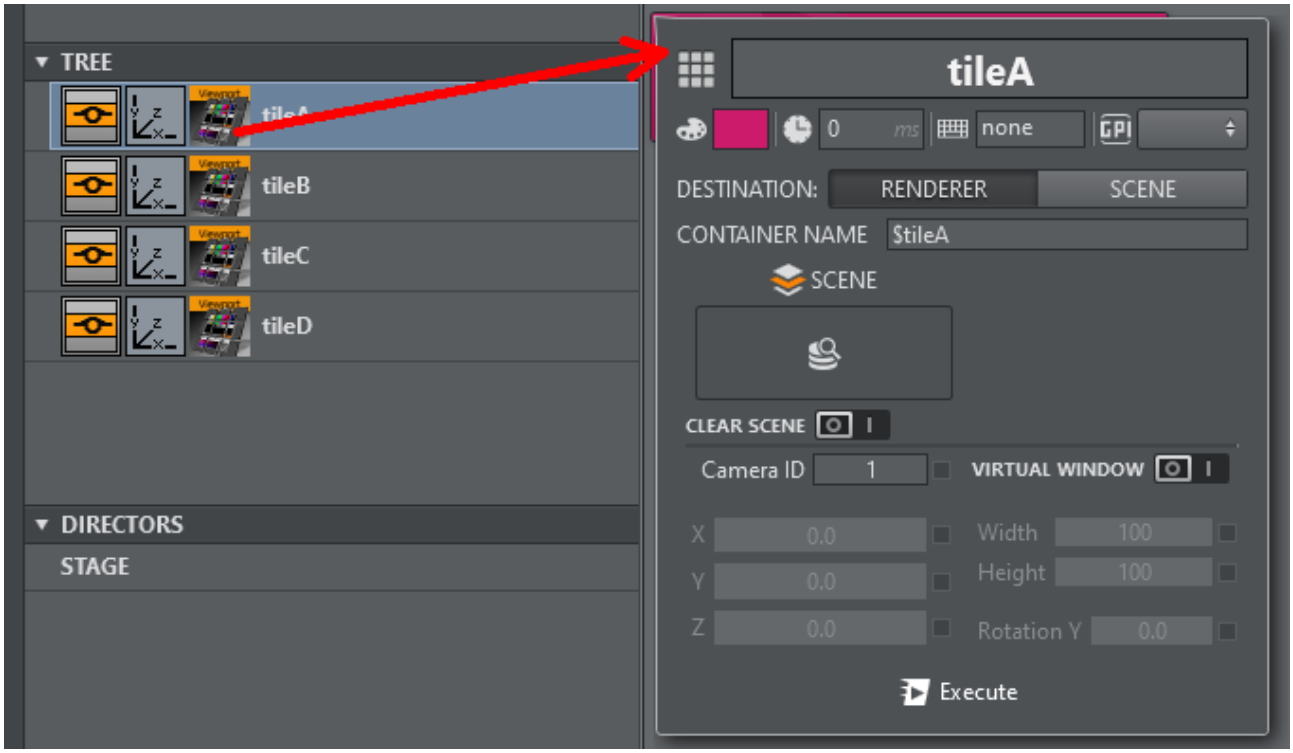
Linked actions are useful for controlling different scenes with the same tree structure, for example.

	Right-click on the panel and select **Create Linked Action** (or **ALT + drag** on an action box).

	A new action box, cloned from the original one, is displayed. Both boxes have the **linked** icon at the top left corner.
	The linked icon also appears on the box saved in the favorite area, if applicable.
	Any changes that are now made to one of the related boxes affects them all, until the clones are removed.

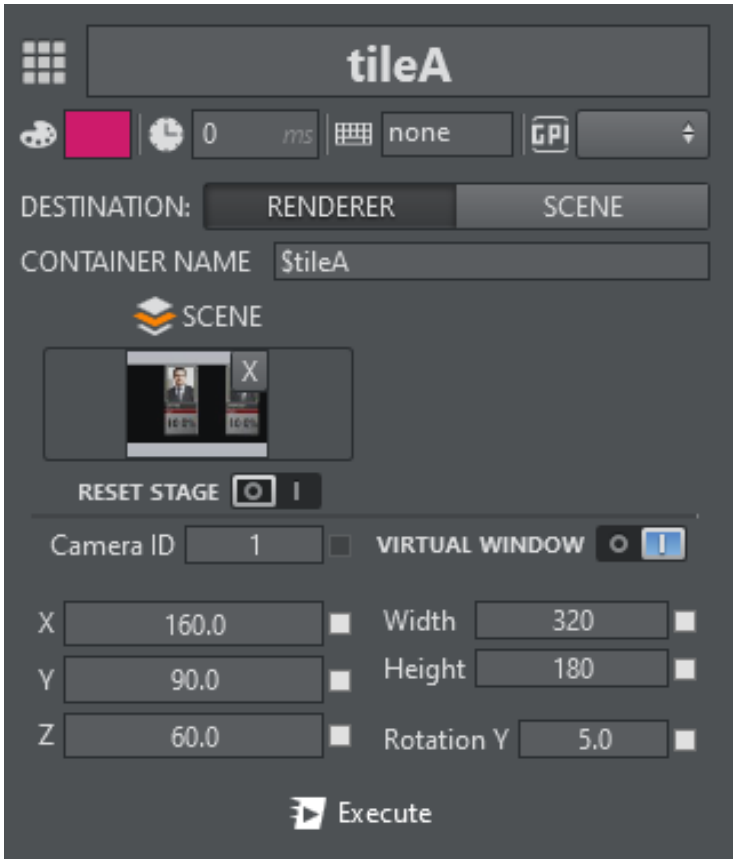
Copyright © 2025 Vizrt

Viewport



Drag and drop a Viewport container plug-in from the tree view into the action canvas to create a **Viewport Action**.

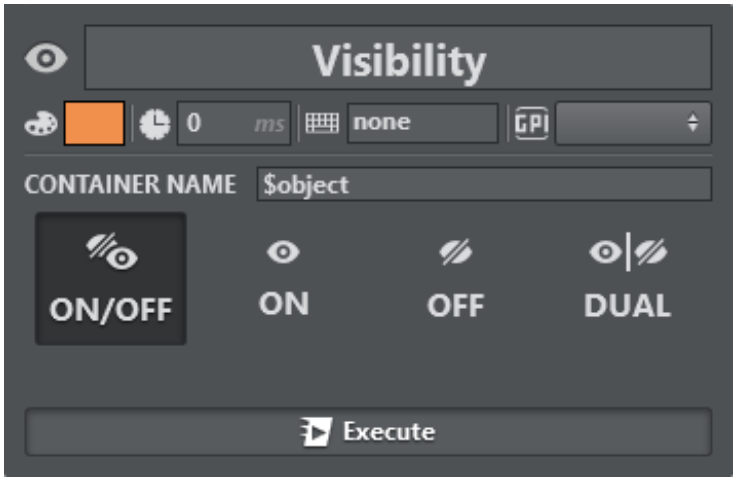
Copyright © 2025 Vizrt



Specify the **Scene** to be loaded to be loaded into the viewport or **Clear Scene** to clear the currently loaded scene of the viewport. Select **Reset Stage** to reset the stage on scene load. **Camera ID** selects the camera to be used within the viewport scene.

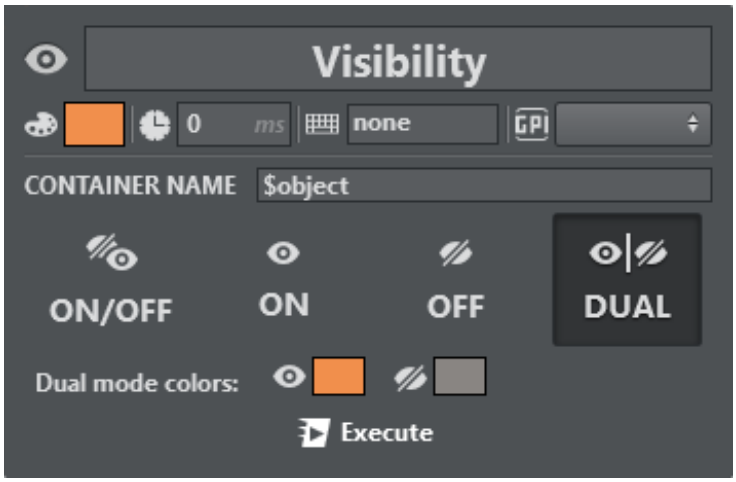
When **Virtual Window** is checked the selected **position**, **rotation** and **width** and **height** are sent to the viewport plug-in.

Visibility



Copyright © 2025 Vizrt

Use the visibility menu to toggle a container on or off. The container can be set to **ON**, **OFF**, **DUAL** or **toggled** (switched to the opposite of the current state).



When **DUAL** mode is selected, left clicking **Execute** switches the container on and right clicking switches the container off. In the **Dual mode colors** section, two distinct colors can be defined for on and off states in order to make the state of the container more visible.

See Also

- Viz Artist User Guide for more information about Scene types.

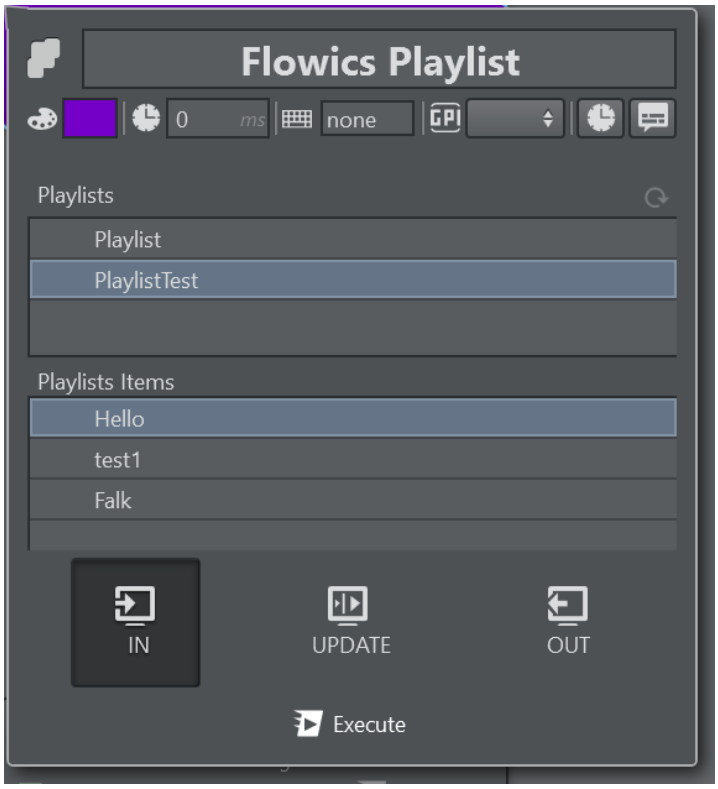
Copyright © 2025 Vizrt

5.4.3 Other Action Types

Right-clicking in the main area makes a menu appear where you can create the following:

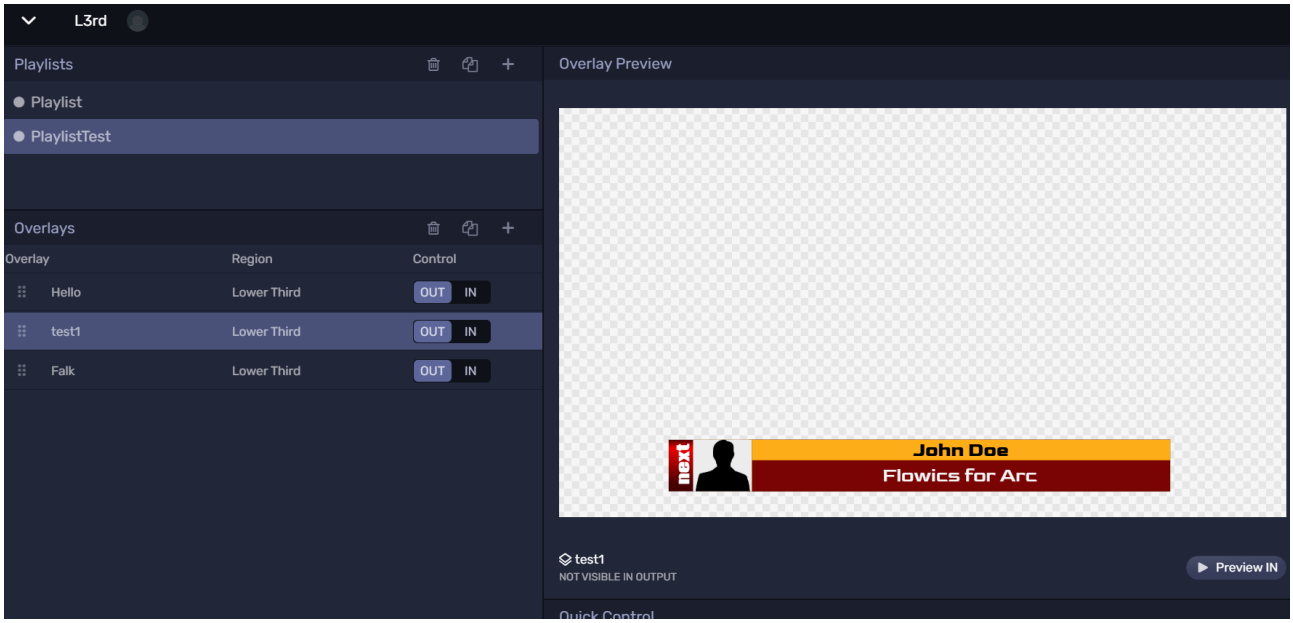
- Flowics Playlist
- Multizone Chroma Key
- NDI
- Scene Loader
- Unreal Scene Loader
    - Using the Browser to Select a Project
- Tecnopoint
- Telemetrics
- Utah
- Viz Clip
- Vinten Control
    - To Import a Vinten Shot

Flowics Playlist

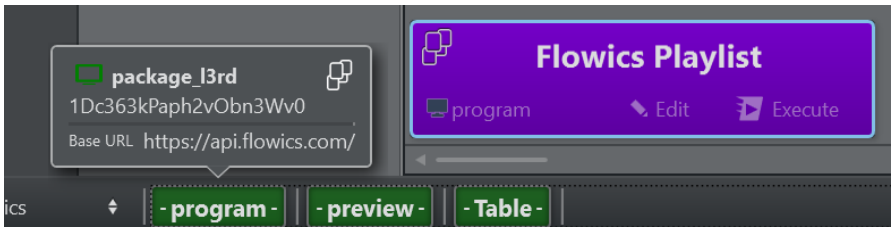


A Flowics Playlist action is used to play out or update a Flowics playlist item. The action can only animate in the playlist element, update it or animate it out. It is not possible to control the content of the playlist items. The graphics package needs to be published as **Rundown Control**.

Copyright © 2025 Vizrt



Make sure the action uses the correct channel containing the Flowics engine pointing to the respective Package.



Copyright © 2025 Vizrt

Multizone Chroma Key



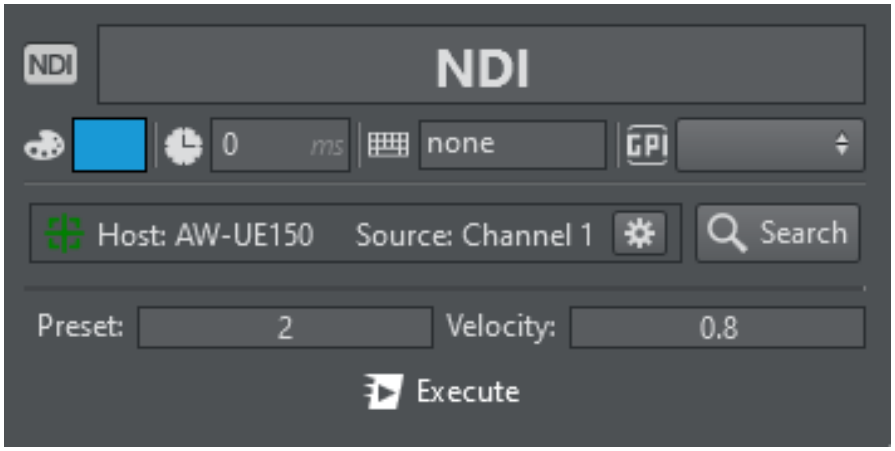
Copyright © 2025 Vizrt

The Multi-zone Chroma key is often used in outdoor sporting event broadcasts, such as sports productions where more than one color needs to be keyed out.

Like the single Chroma Key, it's also possible to set parameters such as the color of your chroma and other fine tuning parameters like **U**, **V**, **Blend** etc. in Multizone Chroma Key.

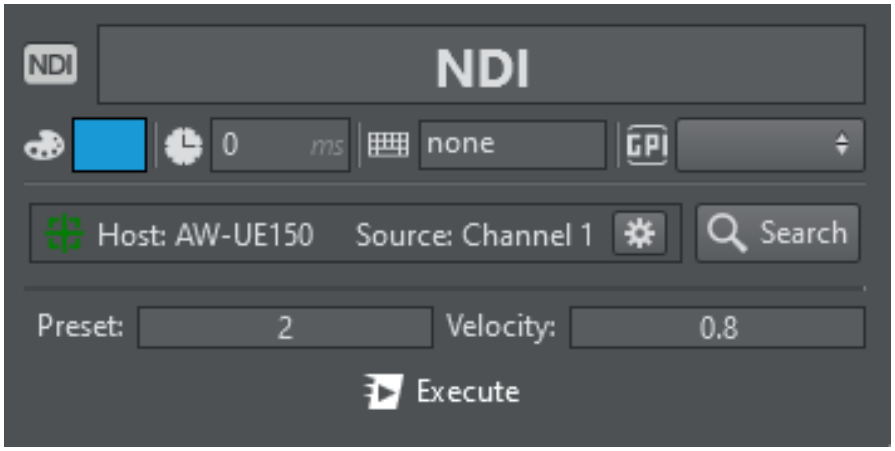
Altitude and **Height** parameters can also be set. See Multizone Chroma Keying for more information on this topic.

NDI



NDI actions allow you to command an NDI camera to go to a certain preset at a configurable velocity.

When creating an NDI action, the configurable content is blocked until an NDI source linked to a controllable camera is selected. This can be achieved by clicking the **Search** button which provides a list of all available NDI sources and let you select those that can be controlled.

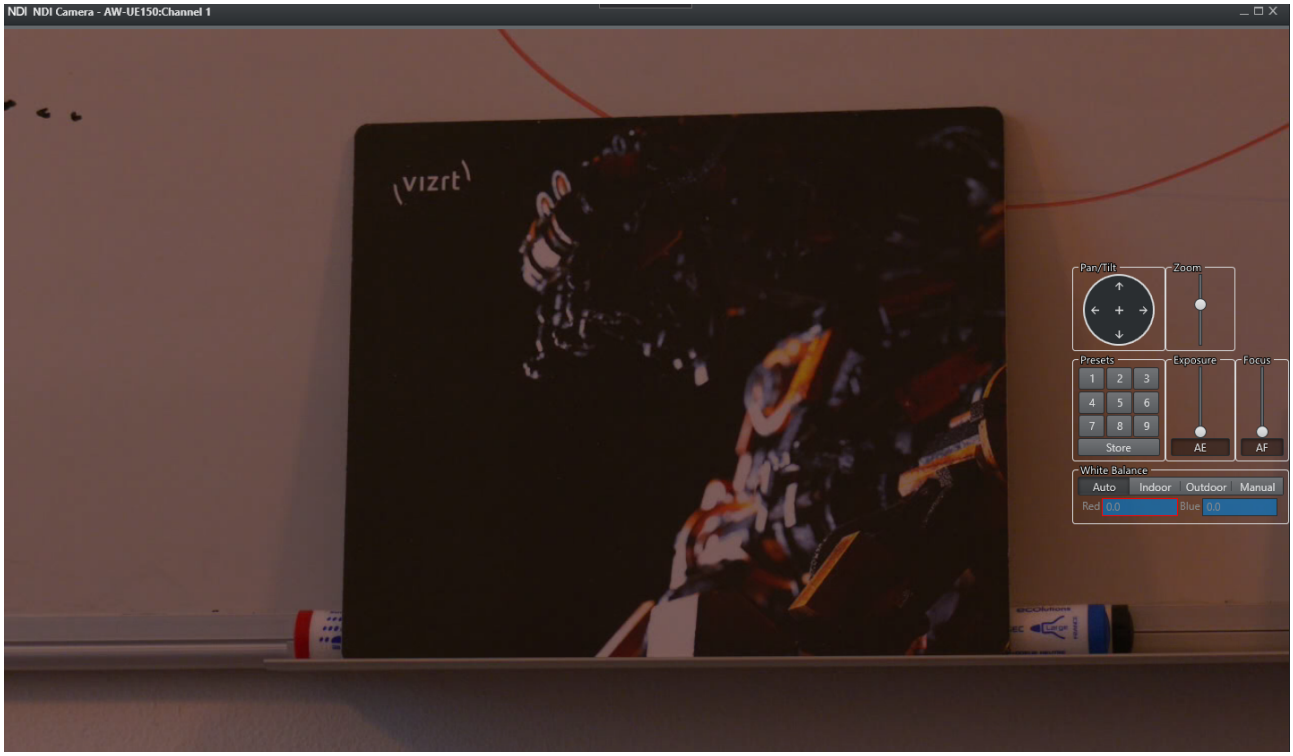


Once a valid source has been selected, the configurable values are unlocked and an UI displaying the connected NDI source is shown. This new interface serves two purposes: to show the NDI source identification and its status; To allow the user to control linked camera and verify / modify its presets.

Clicking on the **Cogwheel** button next to the connected NDI Source opens up a new Window with the NDI Source output and controls that can be used to interact with the camera.

Move the camera (Pan/Tilt), change zoom, exposure (Manual or Automatic), Focus (Manual or Automatic) and white balance, and set and store presets:

Copyright © 2025 Vizrt



When used in conjunction with monogram's ORBITER module, the camera can be operated through the orbiter.

Copyright © 2025 Vizrt

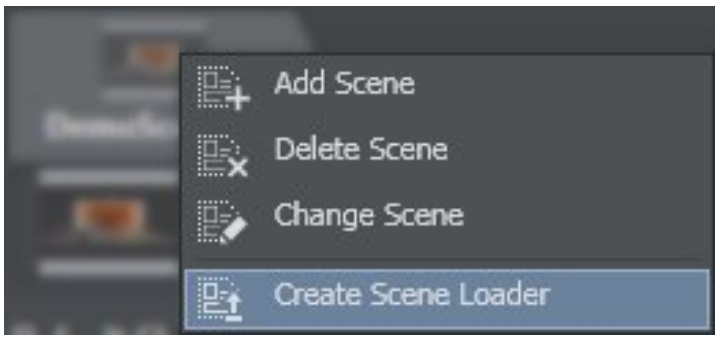
Scene Loader



The Scene Loader lets you configure whether to load a scene in the FRONT_LAYER, MAIN_LAYER, BACK_LAYER and in one of the MATTE or GFX channels (from 1 to 16) at the same time. It's also possible to select the **RESET STAGE** option to load a scene initialized in frame 0.0 on stage. To remove a single scene, press the **X** icon at the top-right of the selected scene. Select **CLEAR LAYER** if you want to explicitly clear a specific layer. Select whether to load the scenes using the *GUID* or the *NAME*.

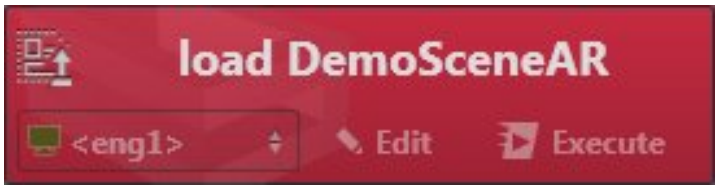
When a scene was selected belonging to a **Concept** and an engine in the selected channel has a different **Concept** configured, the scene loader loads the corresponding scene belonging to that concept.

Create Scene Loader by right-clicking on the desired scene in the panel on the left:

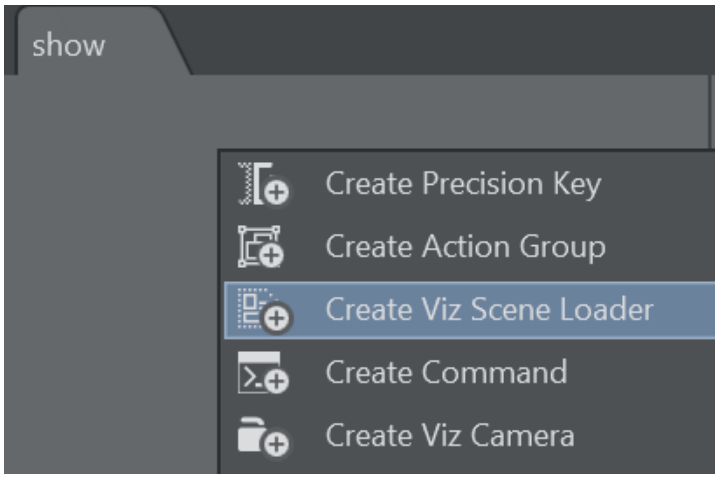


Copyright © 2025 Vizrt

This automatically adds the action to your active action area:



As an alternative, an empty scene loader can be created from the context menu on the action canvas:

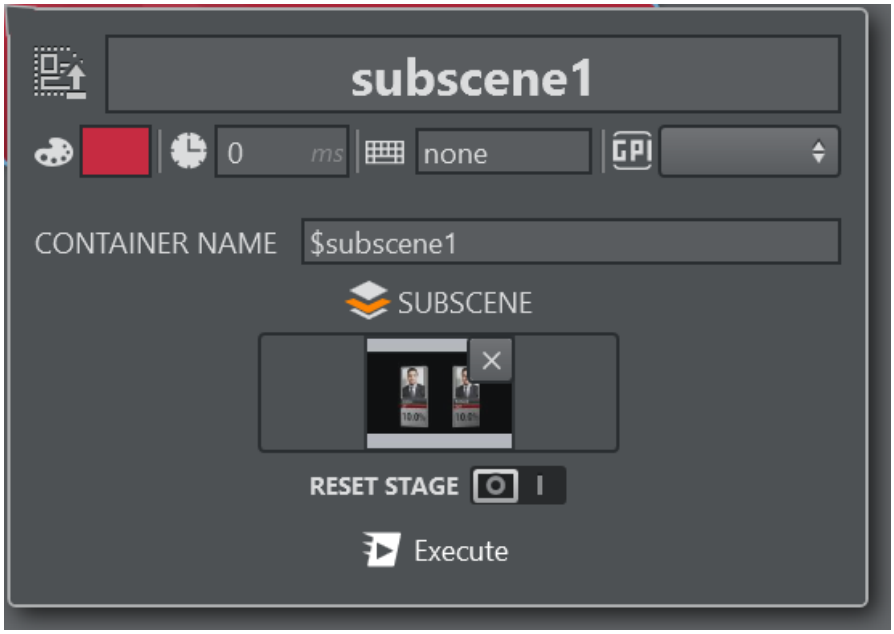


Viz Engine 4 introduces the concept of **SUBSCENES** where entire scenes can be loaded into a container in the scene tree. When loading a scene containing a **SUBSCENE** plugin, it is displayed in Viz Arc's the tree view.



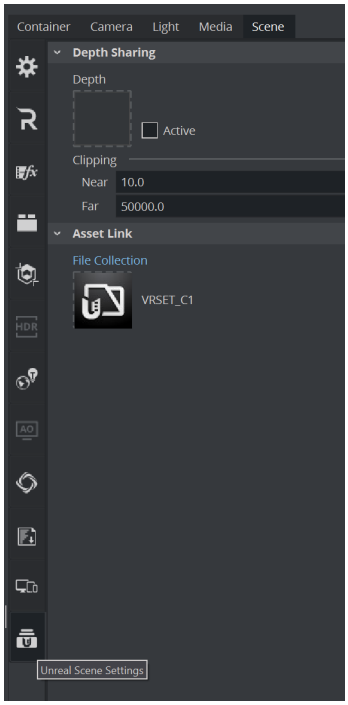
Drag and drop the subscene icon onto the action canvas to create s subscene loader action.

Copyright © 2025 Vizrt



The scene loader can be used to load different scenes in the subscene, clear the scene and reset the subscene director's on load. Under **CONTAINER NAME** the container path of the subscene can be changed manually.

The scene loader can also set the Unreal File Collection associated to a Viz Engine scene

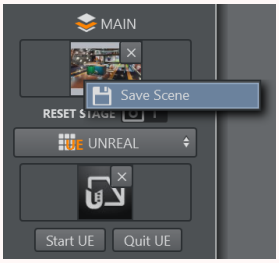


Copyright © 2025 Vizrt



From the drop down menu select the **UNREAL** item and select a File Collection browsing the Graphic Hub. When changing the File Collection, make sure to close the Unreal Engine using the **Quit UE** button, start the new File Collection using the **Start UE** button.

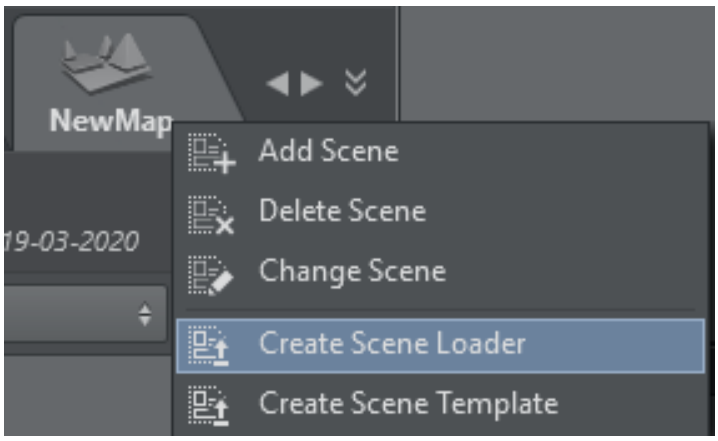
Important: Please note that the Viz Engine will automatically launch the File Collection that has been saved with the scene. Make sure you save the Viz Scene when you want to permanently change the File Collection within a Viz Scene.
In case the respective scene has been loaded in the main layer and it is associated to the scene loader, you can use the context menu to save the scene.



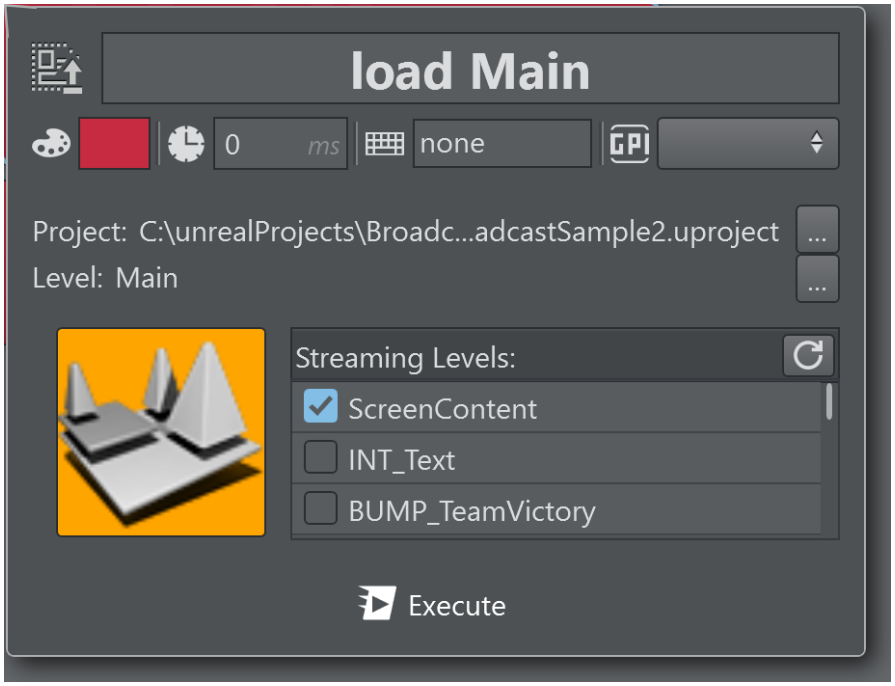
Copyright © 2025 Vizrt

Unreal Scene Loader

The Unreal Scene Loader allows you to load a specific unreal map in a given project. The loader can be created either by using the context menu on the action canvas or through the context menu on the scene tabs.

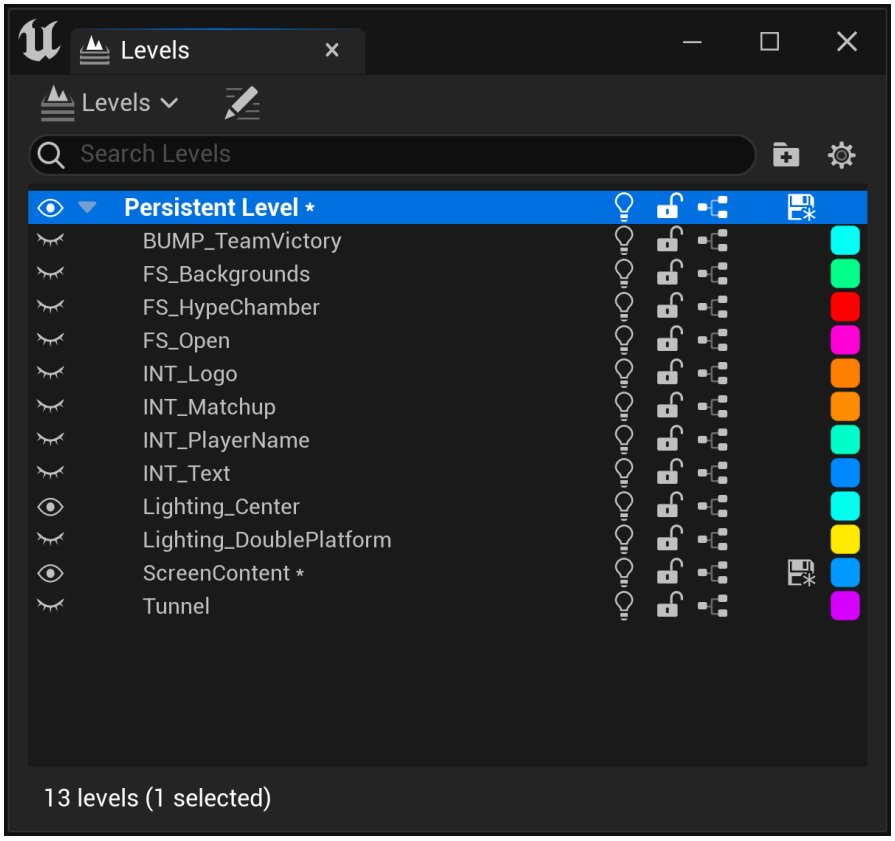


The **Project** indicates the Unreal project while the **Level** indicates the name of the level to be loaded.

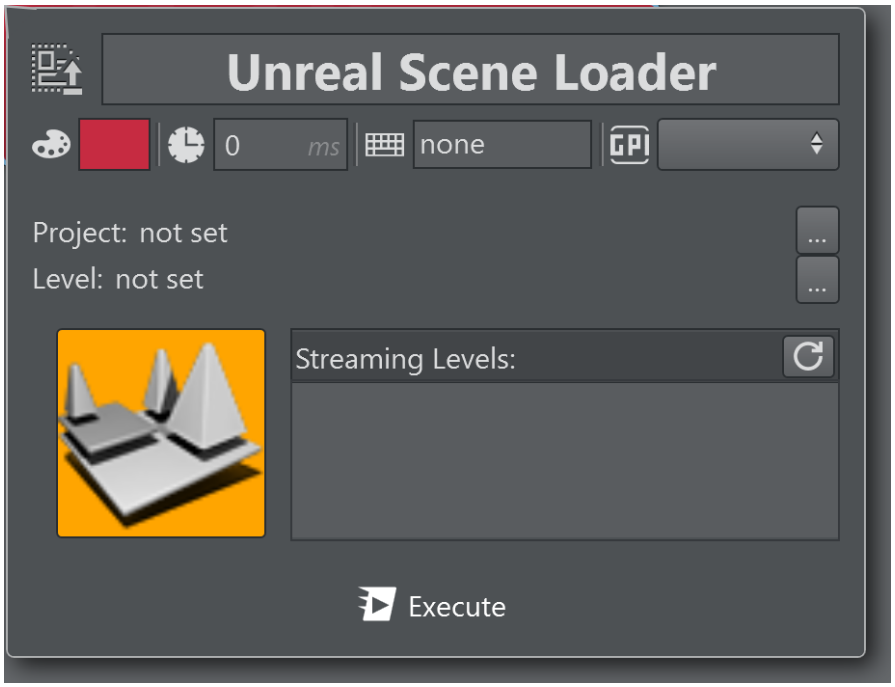


The **Streaming Levels** show a list of the streaming levels that are shown when the scene loader executes.

Copyright © 2025 Vizrt



When creating a scene loader from the action canvas, click on **Select Unreal Level** to open the browser.

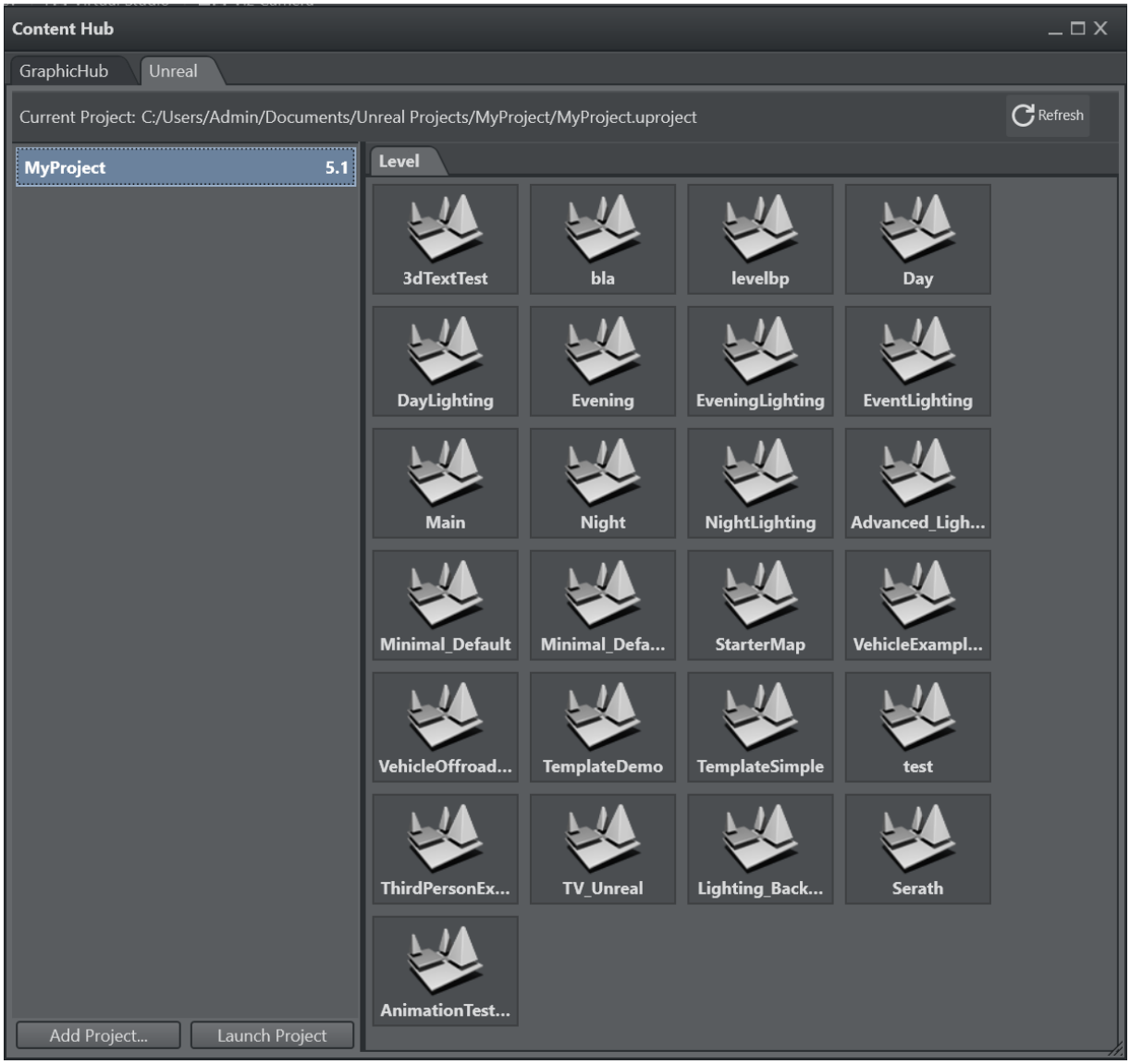


Either browse the unreal project and level browser to manually select the *.uproj* and *.umap* files or click the map icon use the Unreal browser as shown in the screenshot below. Make sure to select the project location before selecting the map level as the map is a location relative to the project.

Copyright © 2025 Vizrt

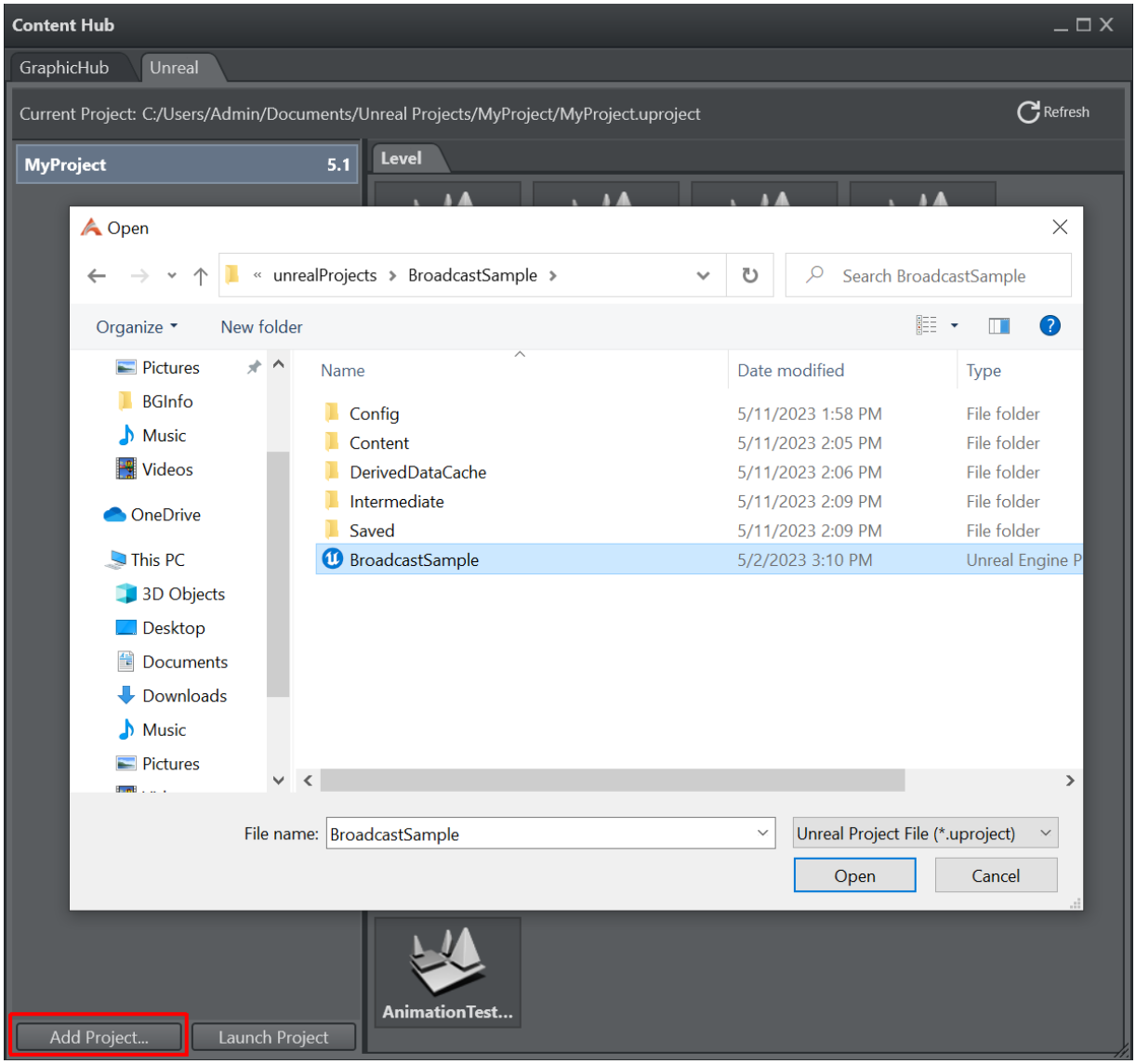
Note: Manual selection of the project and map files implies that the project are stored on a file system that is reachable by the local machine.

The built in browser allows you to browse the content of of the Unreal editing engine which may run on a different machine.



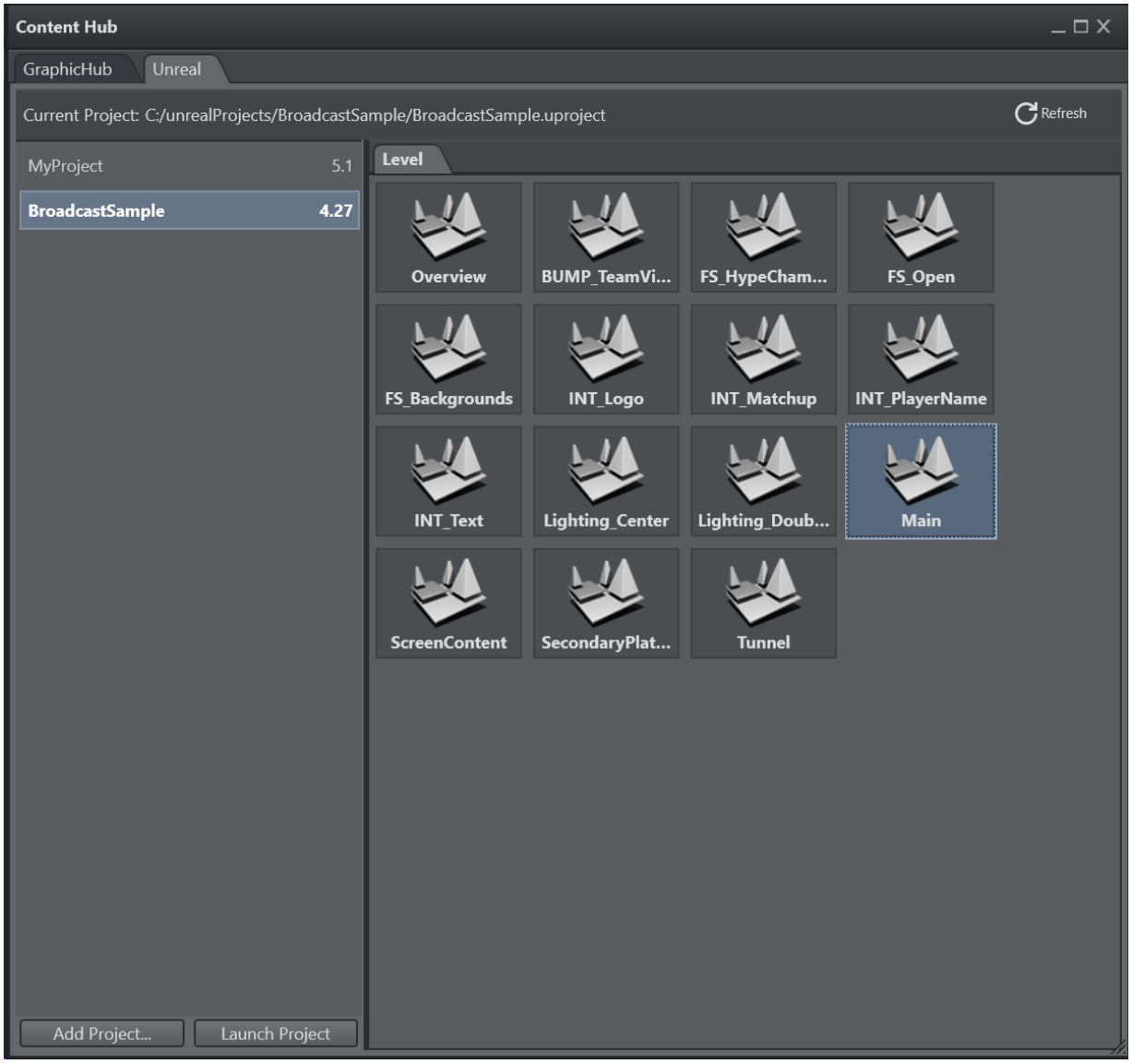
Use the **Add Project...** button to add a project browsing on the local filesystem. The selected project is added to the list of available projects.

Copyright © 2025 Vizrt



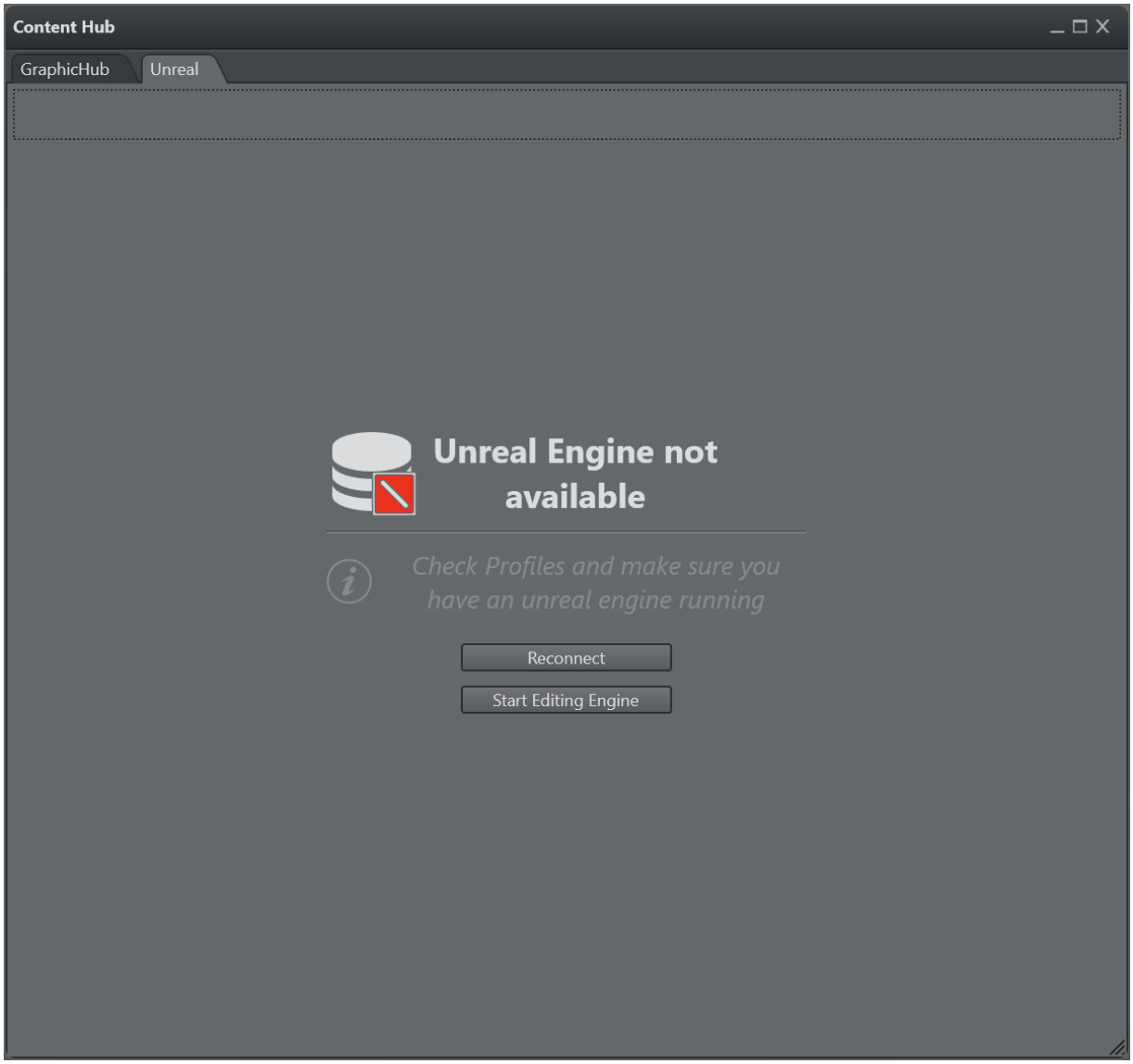
Selecting a new project loads it into the Unreal Engine and displays the available levels. This process may take some time depending on the complexity of the project.

Copyright © 2025 Vizrt



In case the editing Unreal Engine is not running when the browser is opened, you can launch it by clicking the **Start Editing Engine**, this launches the editing engine through the Viz Unreal Launcher. Once the engine is up and running it automatically displays the available projects and levels.

Copyright © 2025 Vizrt



Info: The browser only works when the Unreal **Editing Engine** is in game mode.

Using the Browser to Select a Project

By double clicking a project, the Unreal Engine is restarted with the respective project. The browser only shows the levels of the currently loaded project.

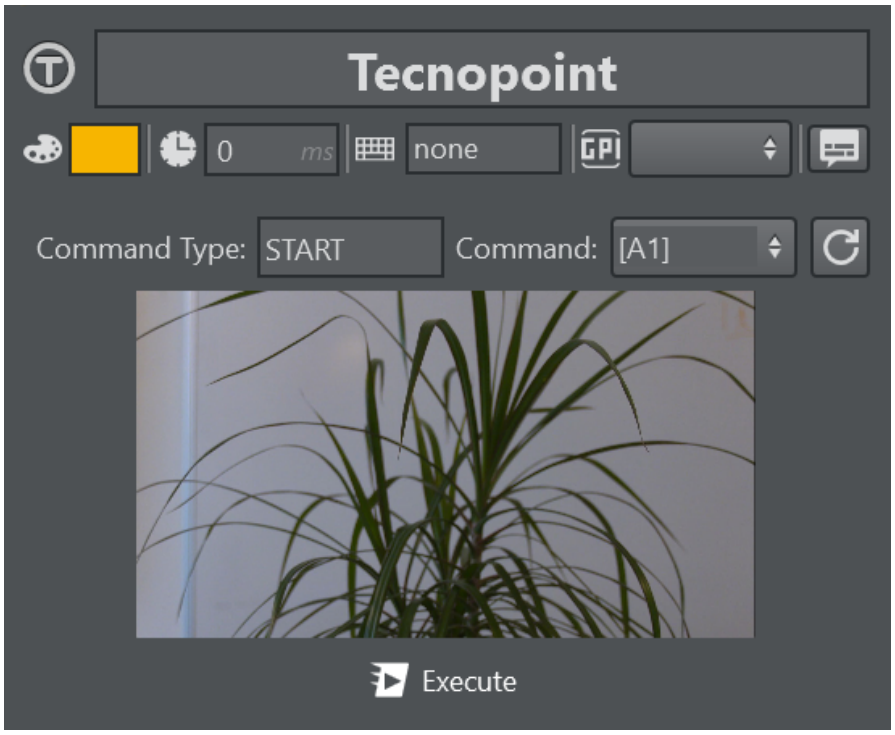
- Click **Refresh** to refresh the layers when the Unreal Engine has been restarted with a different project.
- Double click on one of the layers to confirm the selection.

Note: The unreal scene loader requires the Unreal Loader Service to be properly installed and configured. Whenever a level is loaded that is not part of the current project, the Unreal Engine is restarted.

Copyright © 2025 Vizrt

Tecnopoint

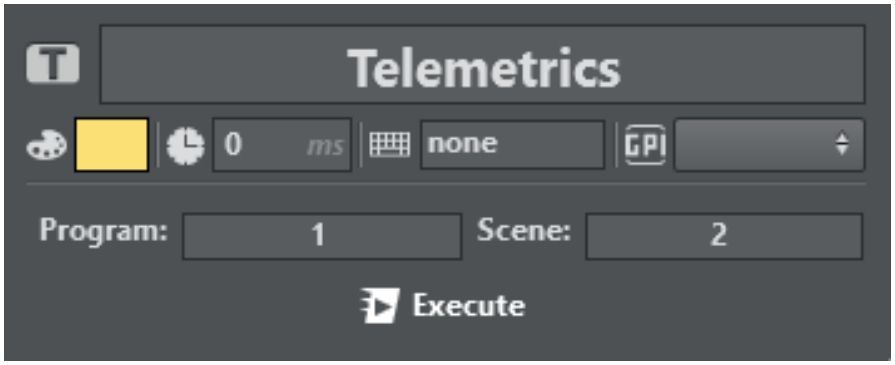
Tecnopoint lets you control a robotic camera using the Tecnopoint TuningS server.



Command Type and **Command** determine the desired shot position of the robotic camera and shot type.

Telemetrics

Telemetrics lets you control a robotic camera using the Telemetrics protocol.

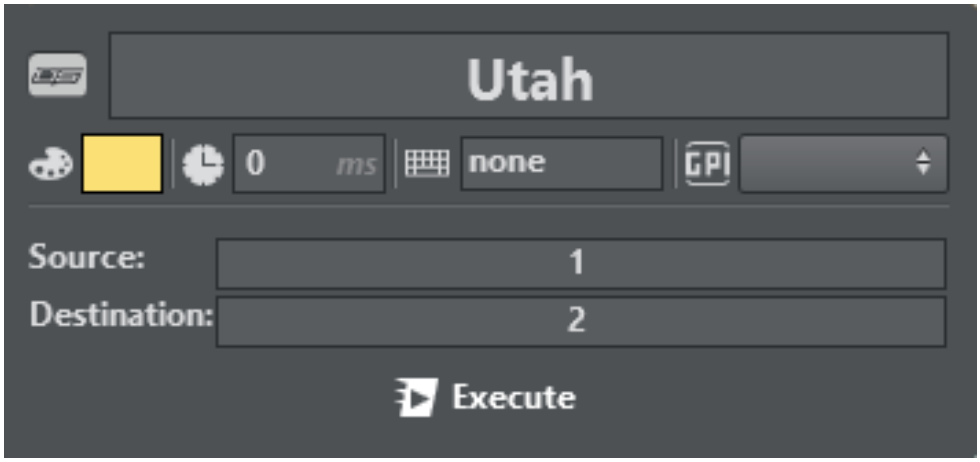


Program and **Scene** determine the desired shot position of the robotic camera.

Utah

Utah is used to route SDI video signals through a Utah Scientific router.

Copyright © 2025 Vizrt



Source and **Destination** determine which source ID is to be mapped to which destination ID.

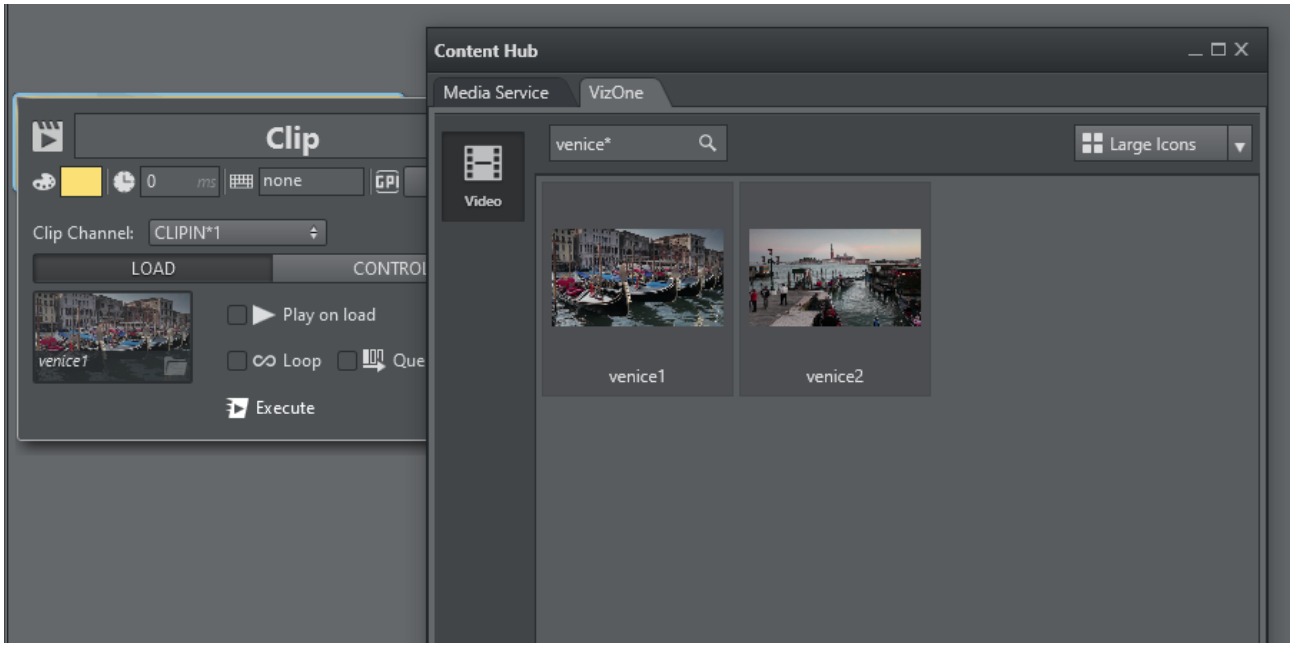
Viz Clip

Viz Clip is used for managing Viz Engine's ClipIn channel; it has two working modes:

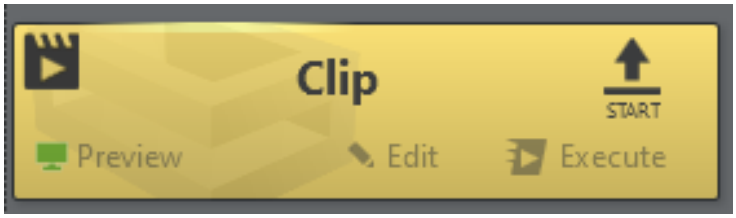
	In the **LOAD** menu, you can select which clip to play (Local file system or Media service) and play options (**Play on load, Loop** and **Queue Clip**). **Queue Clip** lets you set the selected clip to the pending position. It then plays as soon as the currently playing clip finishes.
	Use the **CONTROL** menu to **START**, **STOP**, **CONTINUE** and **PAUSE** the clip**.**

If a Viz One server has been configured, the Clip Action can also browse for clips on Viz One.

Copyright © 2025 Vizrt



The Viz Engines inside the selected channel need to be configured as Engine endpoints in Viz One in order to transfer the clips on the Engines. A progress bar on top of the action indicates the transfer progress.



Copyright © 2025 Vizrt

Vinten Control



Vinten has it's own software and service for controlling cameras. You can use the joystick to move cameras and save the **shot.** It's useful for broadcasters to have the same opening camera animation easily available on a TV program.

Shots are the saved camera movements in Vinten Automation Services software.

Once you have created and saved a shot in Vinten, it's possible to import it into Viz Arc using the Vinten Control menu. There are three timer controls:

- **JUMP TO:** The quickest way to switch from the selected shot to the next shot.
- **MOVE TO:** Configures a time in which it does the shot animation; if the time is too early, it does the animation in the minimum time requested.
- **STOP:** Stops the selected shot.

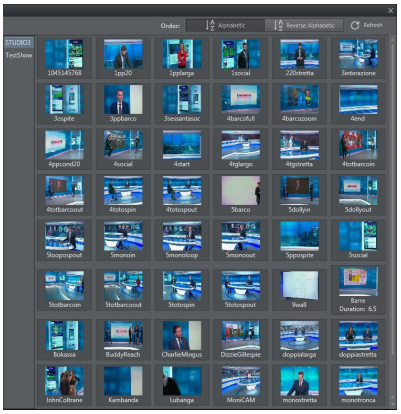
To Import a Vinten Shot

A step-by-step procedure for importing a Vinten Shot into your Viz Arc project follows below:



Select **Create Vinten Control Action** from the context menu to make the panel on the left appear. The shot list is empty.

Copyright © 2025 Vizrt

	To add a Vinten shot, press the **Add Shot Command** button Add Shot Command. A window containing all of the shots saved in the Vinten software (*Vinten Automation Services*) is displayed. It's possible to reorder shots alphabetically or reverse and refresh by pressing the respective buttons at the top right of the window.
	Select your desired shot, and it is automatically added to the action box. You can now assign a time mode between **JUMP** or **MOVE**. It's possible to import multiple shots into the same action box and create a list or keep every shot in a single box.
Add Stop Command	It's also possible to add a stop command by pressing the **Add Stop Command** button. To remove any shot or stop, press the **Trash can** icon next to the desired element.

Copyright © 2025 Vizrt

5.5 Builder Mode

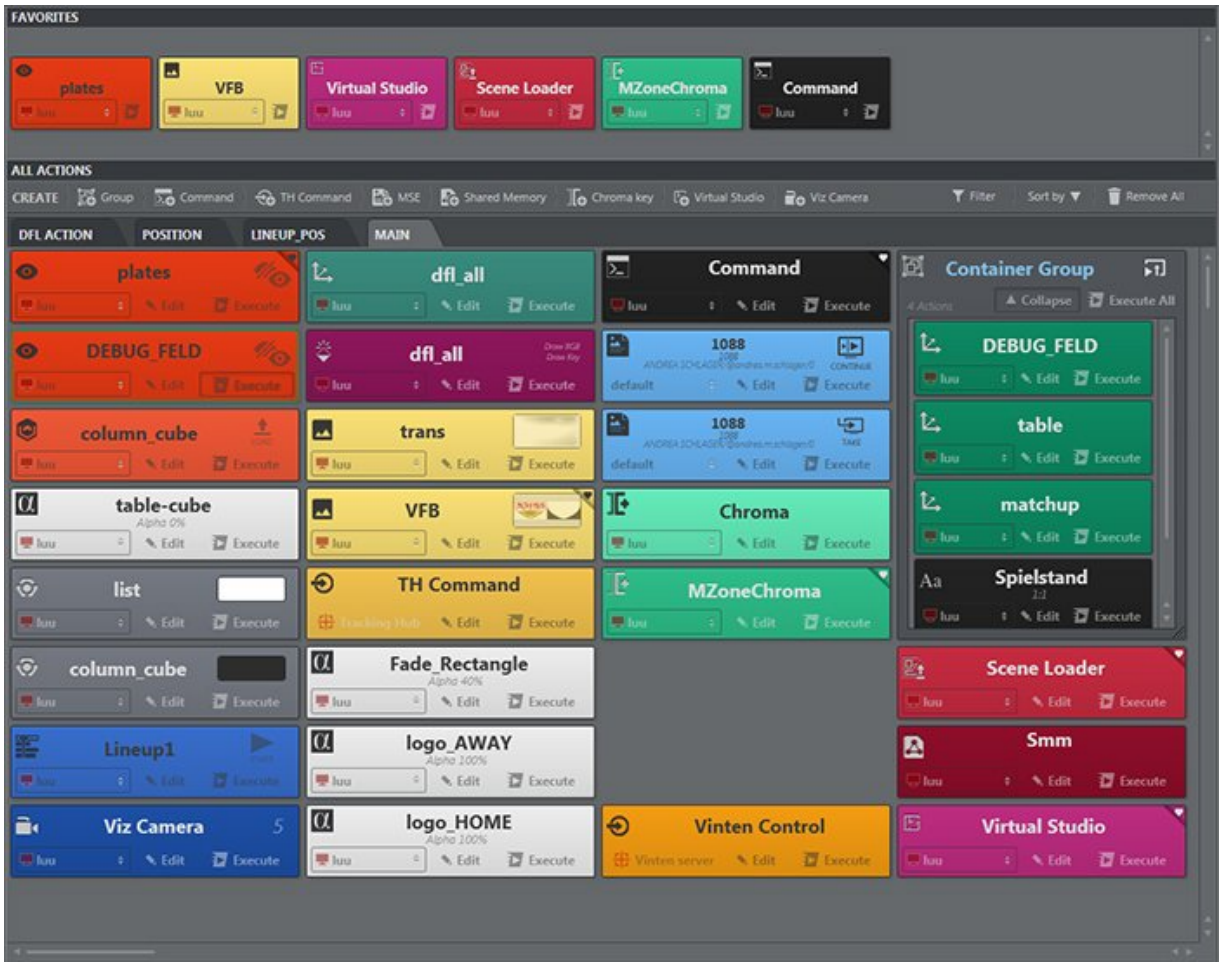
Builder mode lets you build and configure all the actions required for On Air operation.

Builder mode consists of three main views containing a variety of tools for creating projects:

- Actions View
- Set View
- Script View

Copyright © 2025 Vizrt

5.5.1 Actions View



Actions View contains all the actions that are part of your Viz Arc project.

- Actions can be arranged in any way throughout the canvas.
- Easy access to the most used actions can be gained by adding them to the **Favorites** Bar.
- Actions can be filtered for quick access.

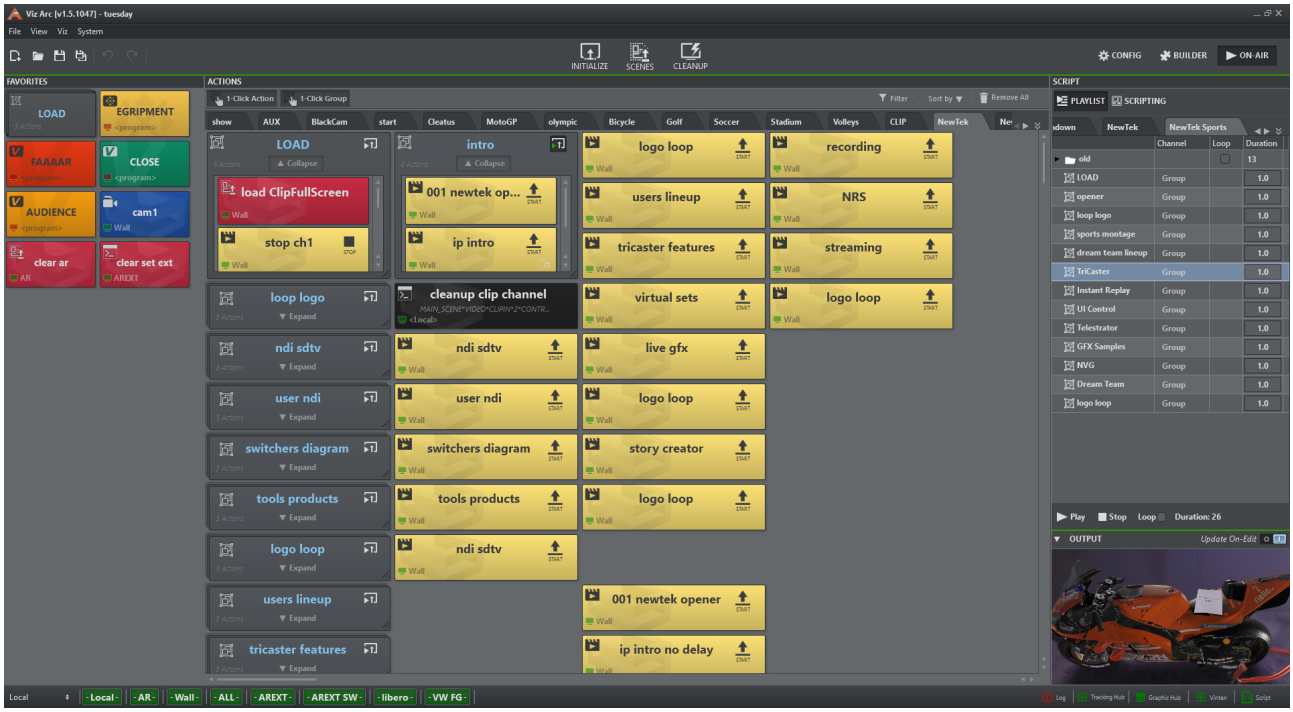
This section covers the following topics and procedures:

- Working with Playlists
  - Creating a Playlist
  - Creating Groups
  - Loop Mode
  - Group Duration
  - Executing an Action in Playlist
  - Renaming an Action in a Playlist
  - Working with Multiple Playlists
- Creating and Editing Scene Actions
  - Scene Actions
  - Project Actions

Copyright © 2025 Vizrt

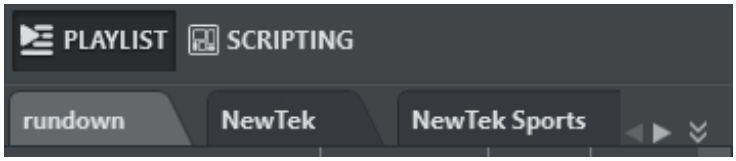
    - To Create Project Actions
    - Editing Actions
- Common Action Tools
    - Channel
    - Edit Mode
    - Execute
- Favorites
    - To Add Actions to Favorites
    - To Remove Actions from Favorites
    - To Remove All Actions from Favorites
    - Searching, Sorting and Removing All Actions
    - Search
    - Sort By
    - Remove All
- Save Action as Data to Viz Pilot

Working with Playlists



Creating a Playlist

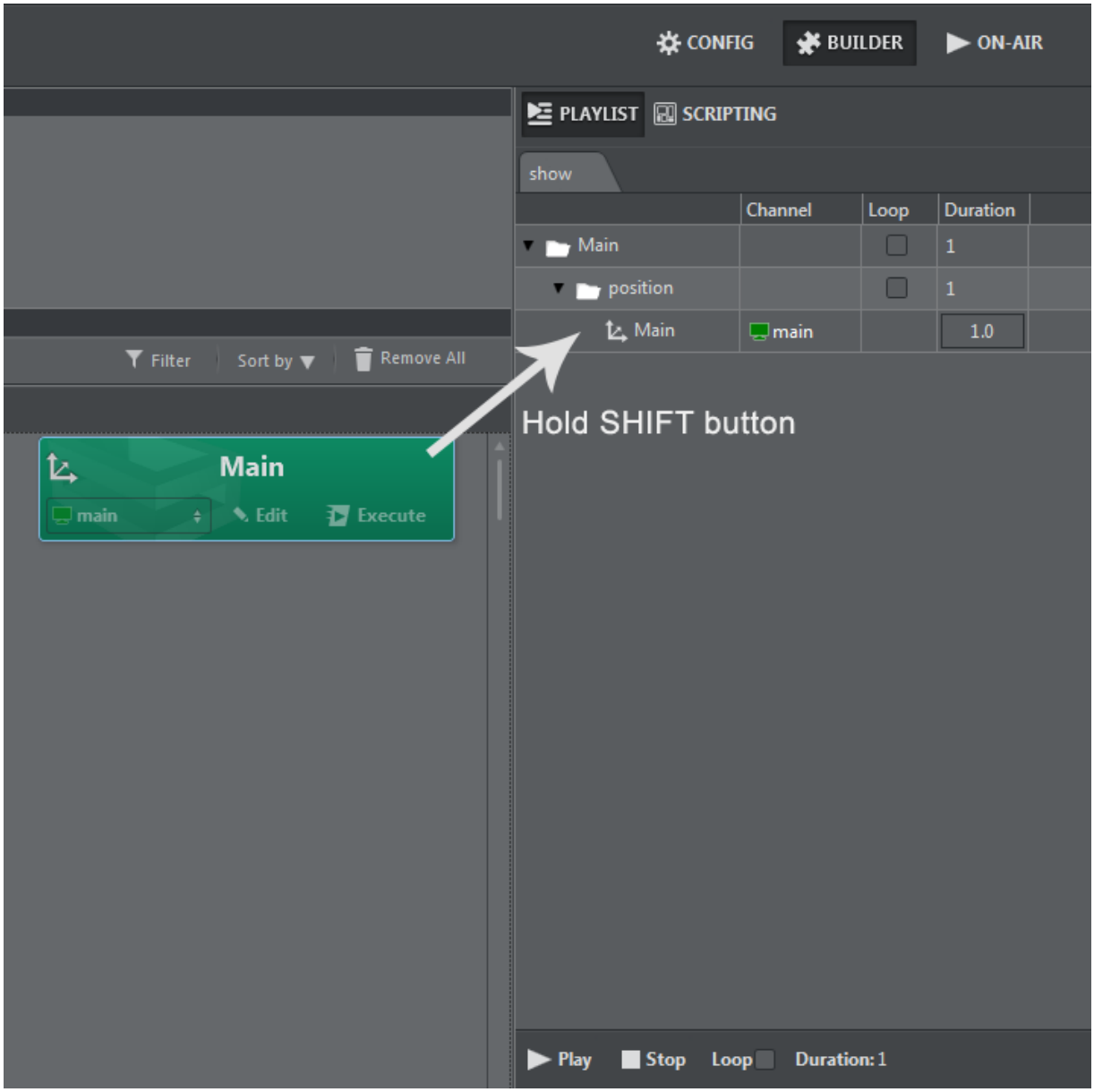
You can create a **PLAYLIST** in the upper right section in **ACTIONS** view. Playlists are useful for arranging actions and creating a rundown.





Import by dragging the desired action into the playlist pane while holding in the **SHIFT** key. Release the action inside the playlist pane to include it in the playlist.

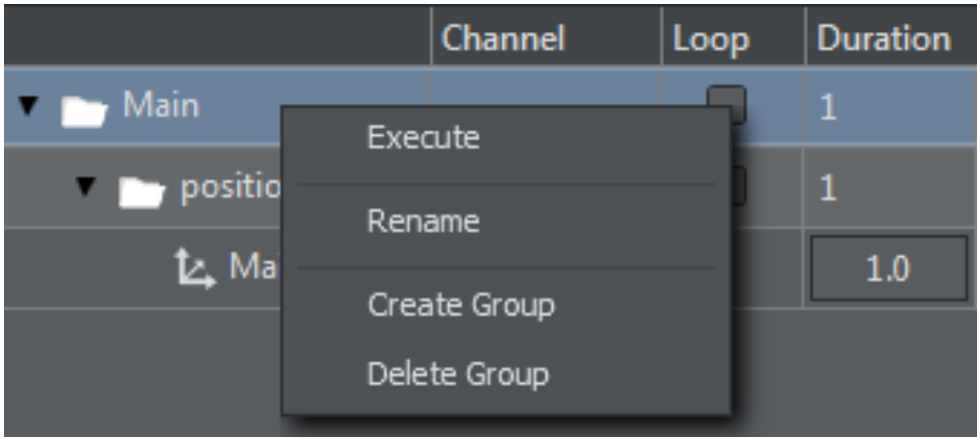
Copyright © 2025 Vizrt



Creating Groups

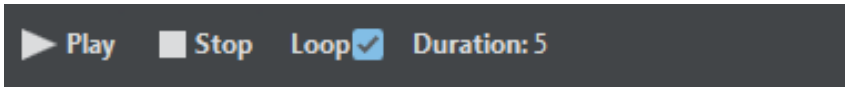
Create groups and sub-groups by right-clicking **Create Group**:

Copyright © 2025 Vizrt



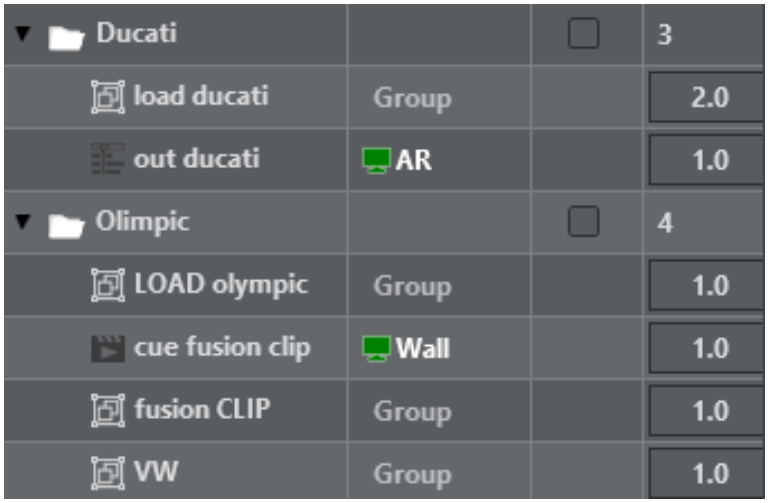
Loop Mode

To set a full playlist in loop mode, enable the main **Loop** selection at the bottom of the playlist pane.



Group Duration

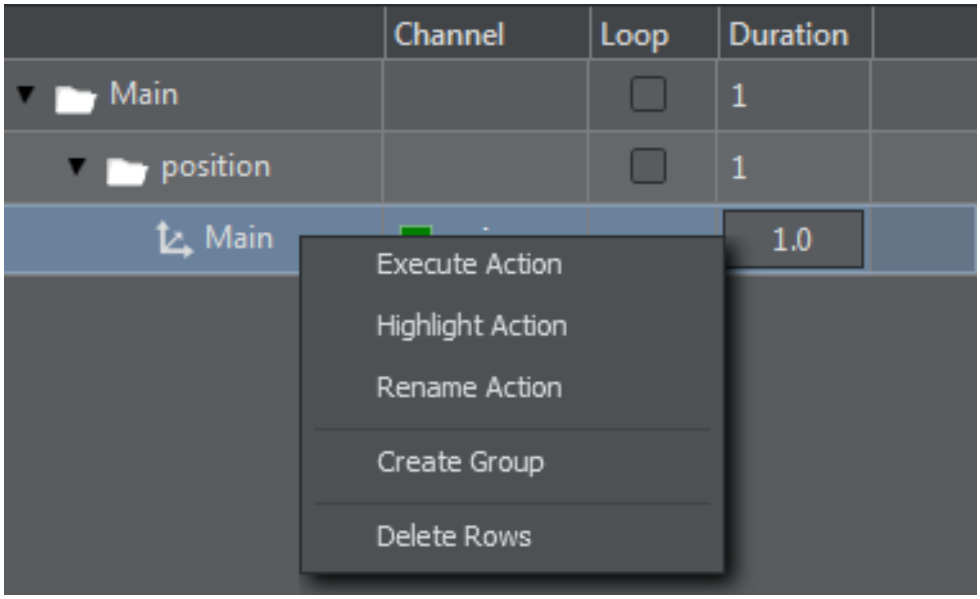
It's possible to specify the duration in seconds of each action (the default is one second). The group's duration is the sum of the duration of all of the actions it contains.



Executing an Action in Playlist

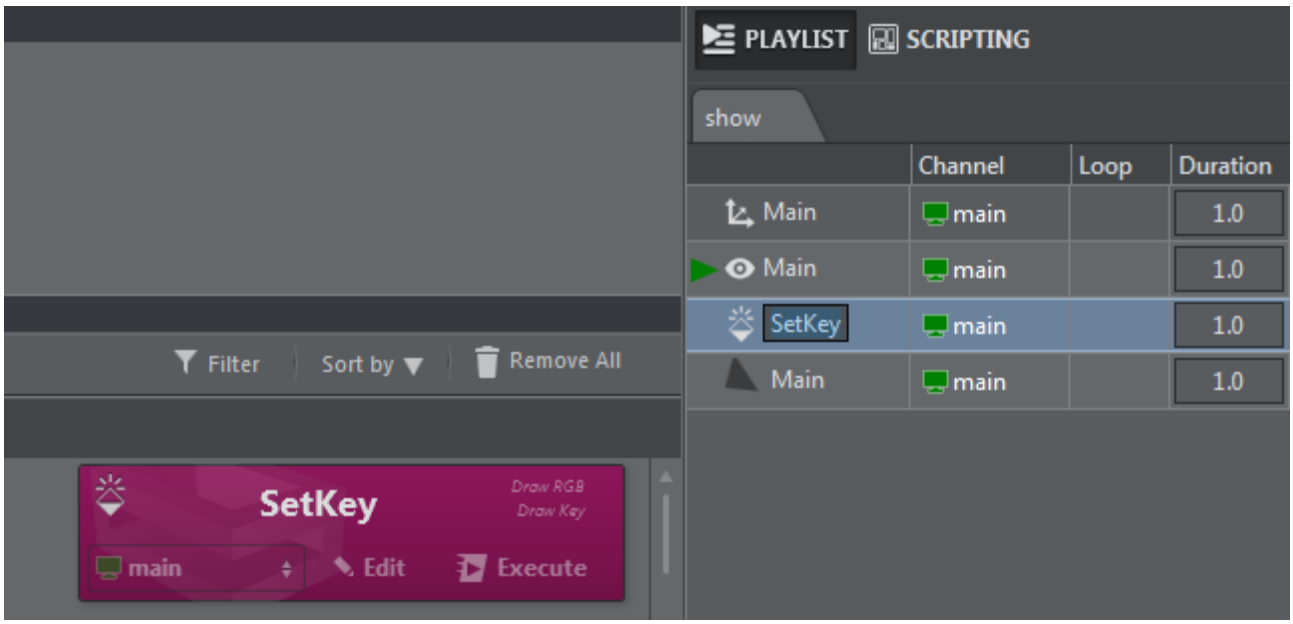
Right-click on on action and select **Execute Action** from the drop down menu:

Copyright © 2025 Vizrt



Renaming an Action in a Playlist

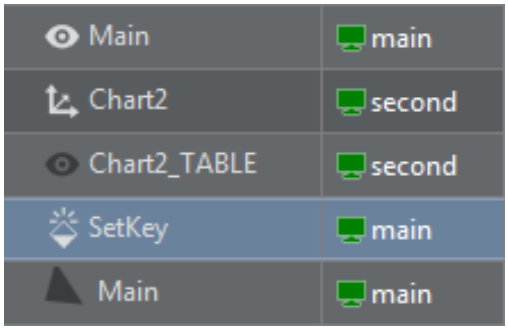
To rename an action, right-click on the selected action and click **Rename Action:**



Note: The action is also renamed in **Actions** view.

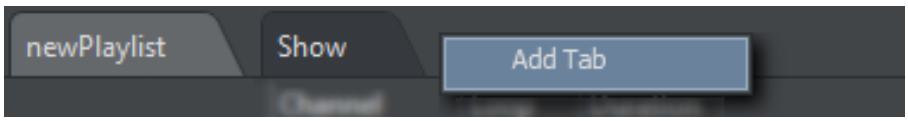
All actions imported in playlist keep their associated channel, so it is possible to send actions to different engines from the same playlist rundown:

Copyright © 2025 Vizrt

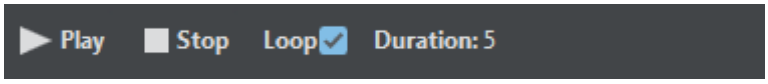


Working with Multiple Playlists

A single project can handle multiple playlists. To work with multiple playlists, right-click inside the **Tab** pane and select **Add Tab**.



Once you are finished creating playlists, you can control the rundown by pressing **PLAY, STOP** or **LOOP**.



Note: The duration value is the total duration in seconds of all the elements in playlist (or the currently active playlist, if a project contains more than one playlist).

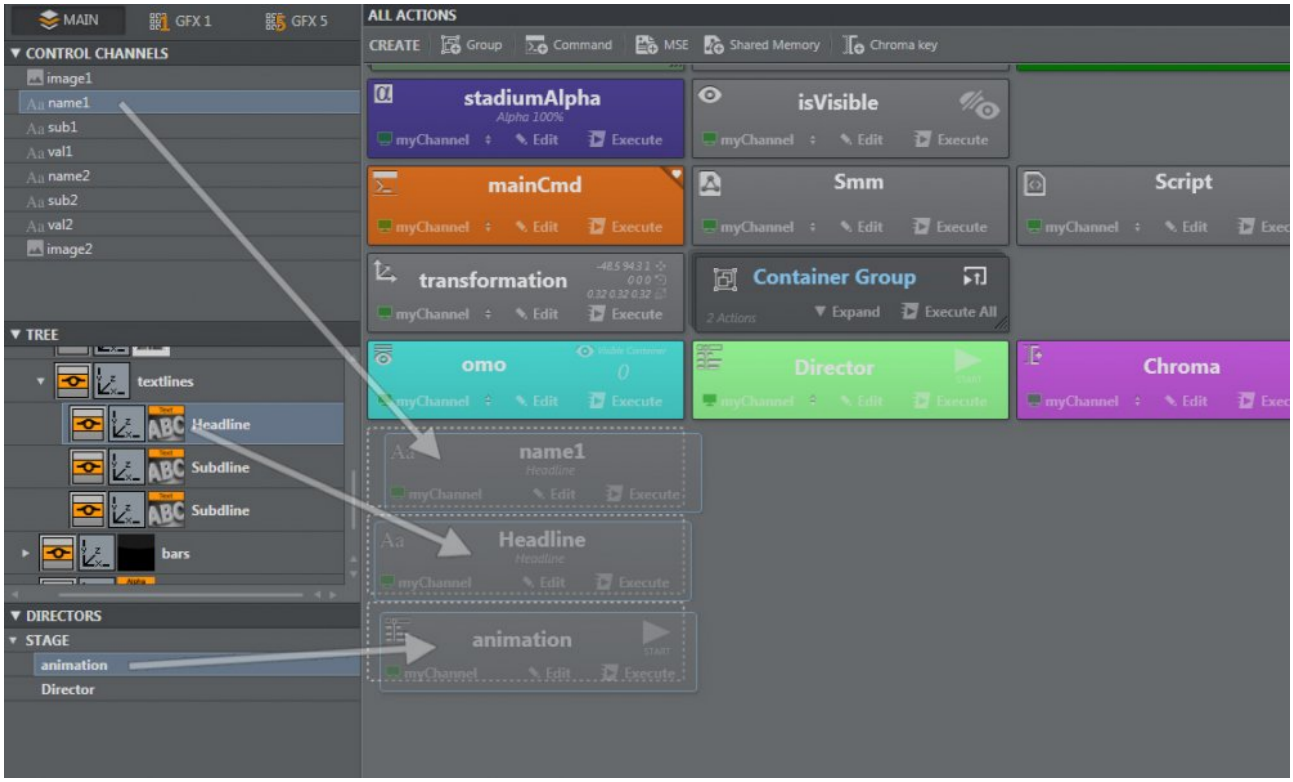
Creating and Editing Scene Actions

Note: All actions share a common set of tools and properties. For more details, see Common Action Tools.

Scene Actions

Before creating scene actions, select the **scene tab** from the scenes panel and load the scene to retrieve properties.

Copyright © 2025 Vizrt



1. Drag and drop any of the supported properties from either the Control Channels, Scene Tree or Directors section in the scene panel to the actions canvas.
2. Enter the desired values in the editing popup.
3. Click anywhere outside the editing popup to save and close your creation.

Project Actions

Project Actions can be created even when no scene is loaded or added into the working project. These action types are not linked to any particular container or scene and can, for example, be used to group actions together or to send Viz commands to a channel.

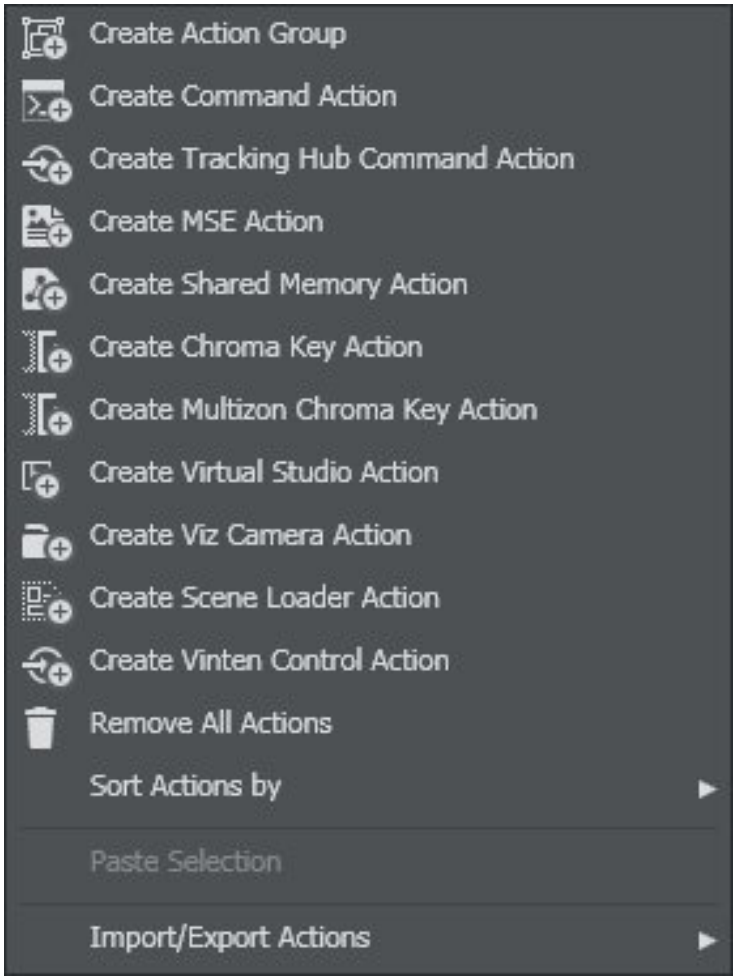
To Create Project Actions



Click on the desired Project Action Types button on the **CREATE** toolbar.

OR

Copyright © 2025 Vizrt



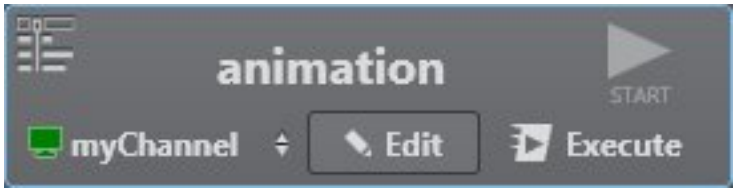
- **Right-click** on the actions canvas.
- **Select** the desired Project Action Types from the context menu.

Info: Other actions can be also found by right-clicking on an empty area in the project. For more information see Other Action Types.

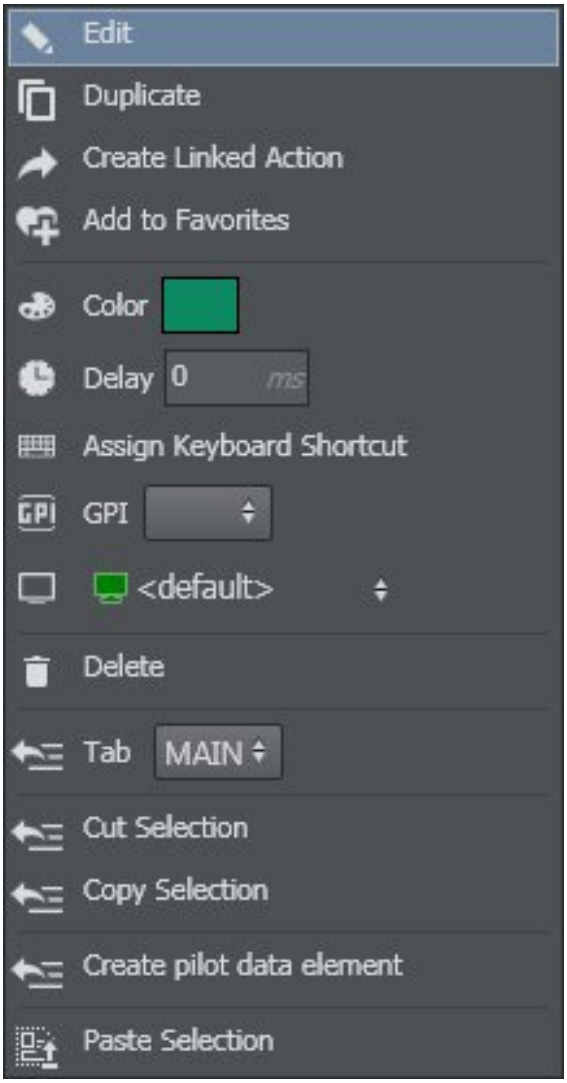
Editing Actions

Any action in a project can be edited to change the logic of the action.

Actions can be edited either:



- By pressing the **Edit** button on any of the actions.
  OR



- By right-clicking on any action and selecting **Edit** from the context menu.

Common Action Tools

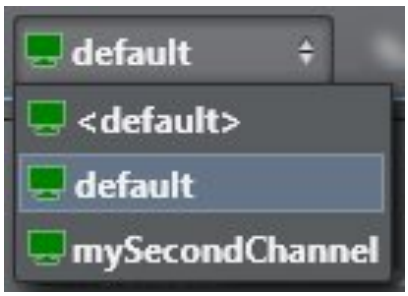
All actions created in Viz Arc share a common set of tools.



1. **Channel:** Selected channel to execute actions.
2. **Edit Mode:** Press the **Edit** button to expand the editing box to modify general action settings.
3. **Execute:** Executes the action.

Copyright © 2025 Vizrt

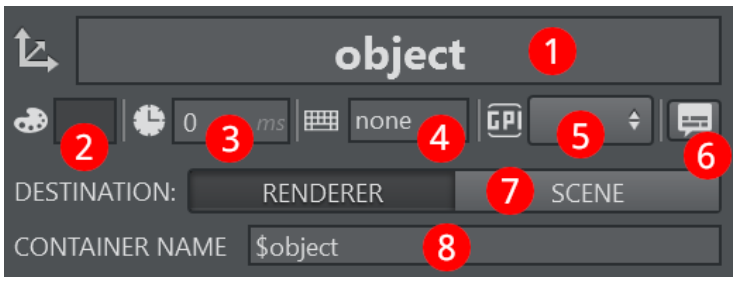
Channel



When executed, all actions send the relevant commands to Viz Engine to manipulate the scenes. These commands are sent to the Engines defined in the selected channel for each action.

Edit Mode

Press the **Edit** button to expand the editing box to modify general action settings.



1. **Action name:** Each action includes a name that can be used to filter actions.
2. **Action Color:** It's possible to assign a color to each action.
3. **Action delay:** Action delay adds an execution delay in milliseconds.
4. **Keyboard shortcut:** Assigned keyboard shortcut.
5. **GPI:** GPI Channel (HI-0/HI-15, LO-0/LO-15).
6. **Tooltip:** Edit the tooltip for the action.
7. **Destination:** Whether the destination for a Viz scene based command should be sent to the renderer or a specific scene.
8. **Control Channel/Container:** The Control Channel or Container path to which the action is linked.

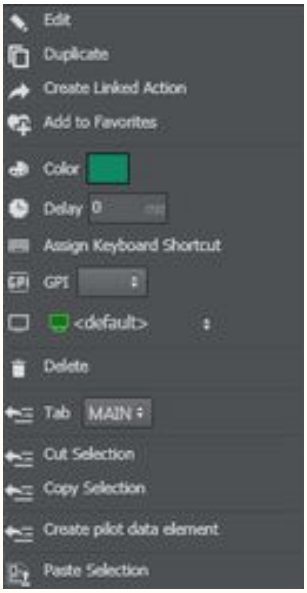
Execute



The **Execute** button lets you send a specific action command to the selected channel.

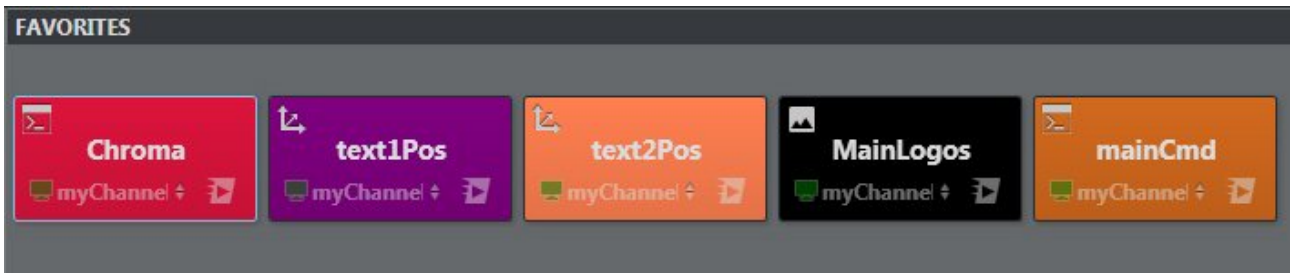
> **Note:** It's also possible to access common action tools through the context menu by right-click on an action box:

Copyright © 2025 Vizrt



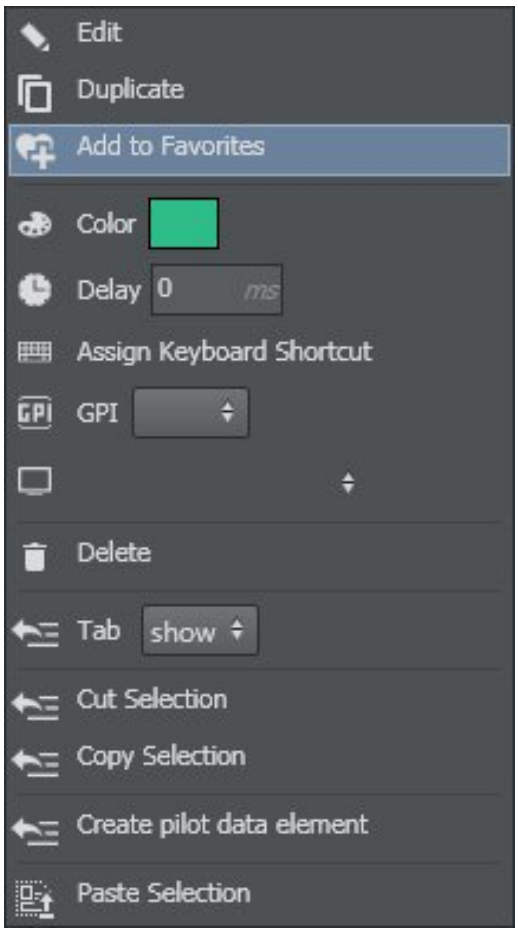
Favorites

Favorites is used for easy access to the most commonly used actions. Favorite actions appear in this section:



Copyright © 2025 Vizrt

To Add Actions to Favorites

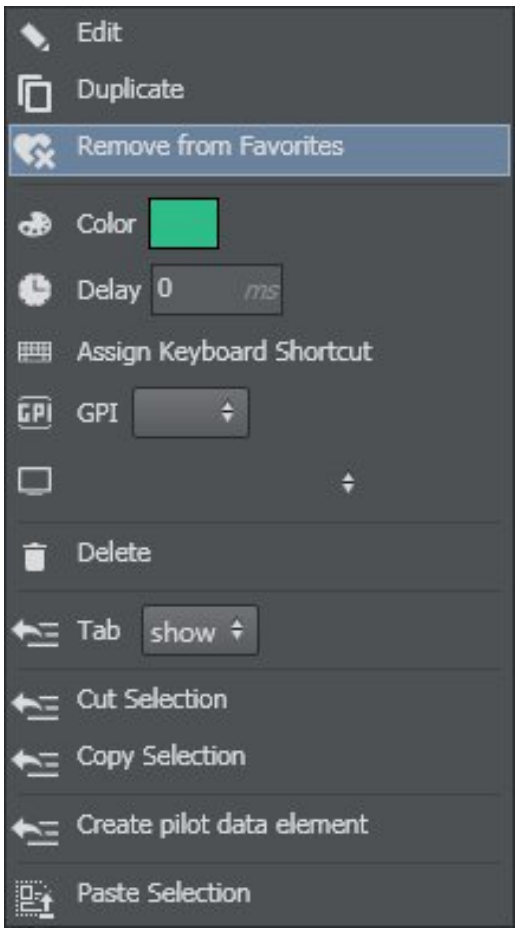


1. Right-click on the selected action or actions.
2. Select **Add to Favorites** from the context menu.

Note: It's possible to add more than one action to the favorites section at the same time. Select all the desired actions and follow the same steps as above.

Copyright © 2025 Vizrt

To Remove Actions from Favorites



1. Right-click on the selected action or actions.
2. Select **Remove from Favorites** from the context menu.

To Remove All Actions from Favorites

It's possible to remove all favorite actions from the favorites section.



1. Right-click anywhere in the favorites section.
2. Select **Clear All Favs** from the context menu.

Searching, Sorting and Removing All Actions

Actions can be searched by name in order to easily locate Actions in a project.

This section describes the different options in the actions tool bar.

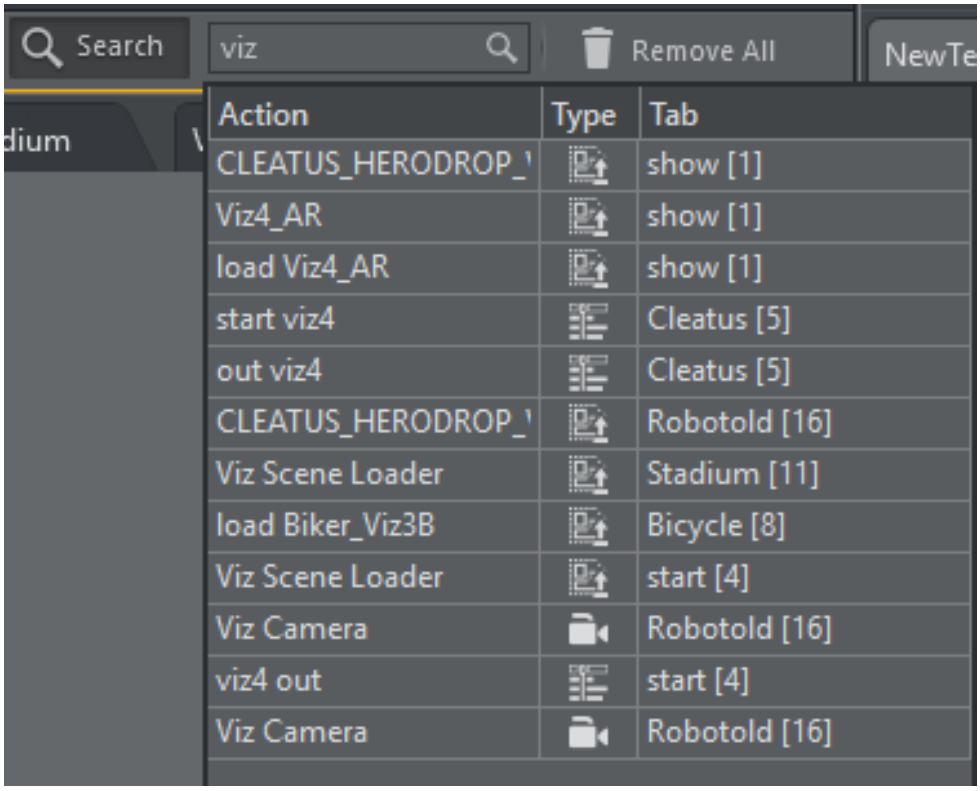


1. **Search:** Search actions by name.

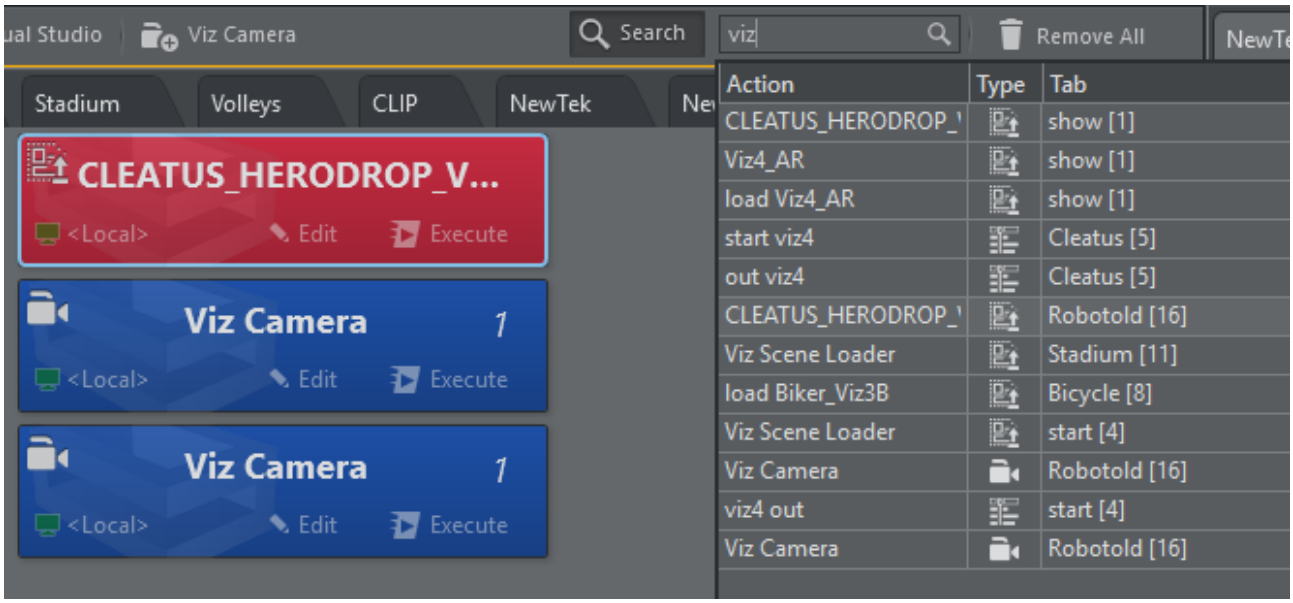
Copyright © 2025 Vizrt

2. **Remove all:** Removes all actions from the project.

Search



Click the **Search** button and a popup will initially show all available actions. Type a string to narrow down the search for an action containing the string in its name. By selecting an action it will be selected and shown on the action canvas.



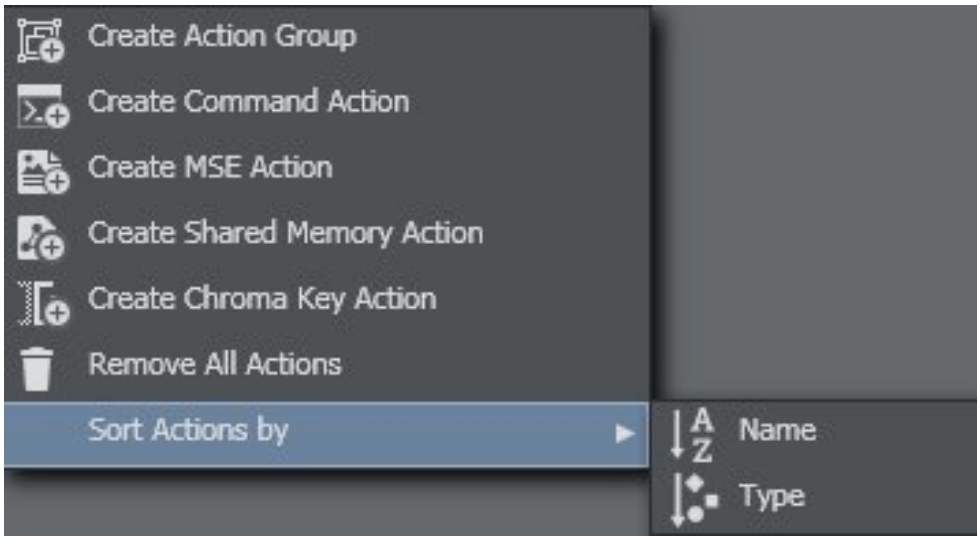
Copyright © 2025 Vizrt

Sort By

It's possible to **sort actions** that are in the project by **name** (ascending) or **type**.



It’s also possible to sort by right-clicking on an empty space in the action area.



Remove All

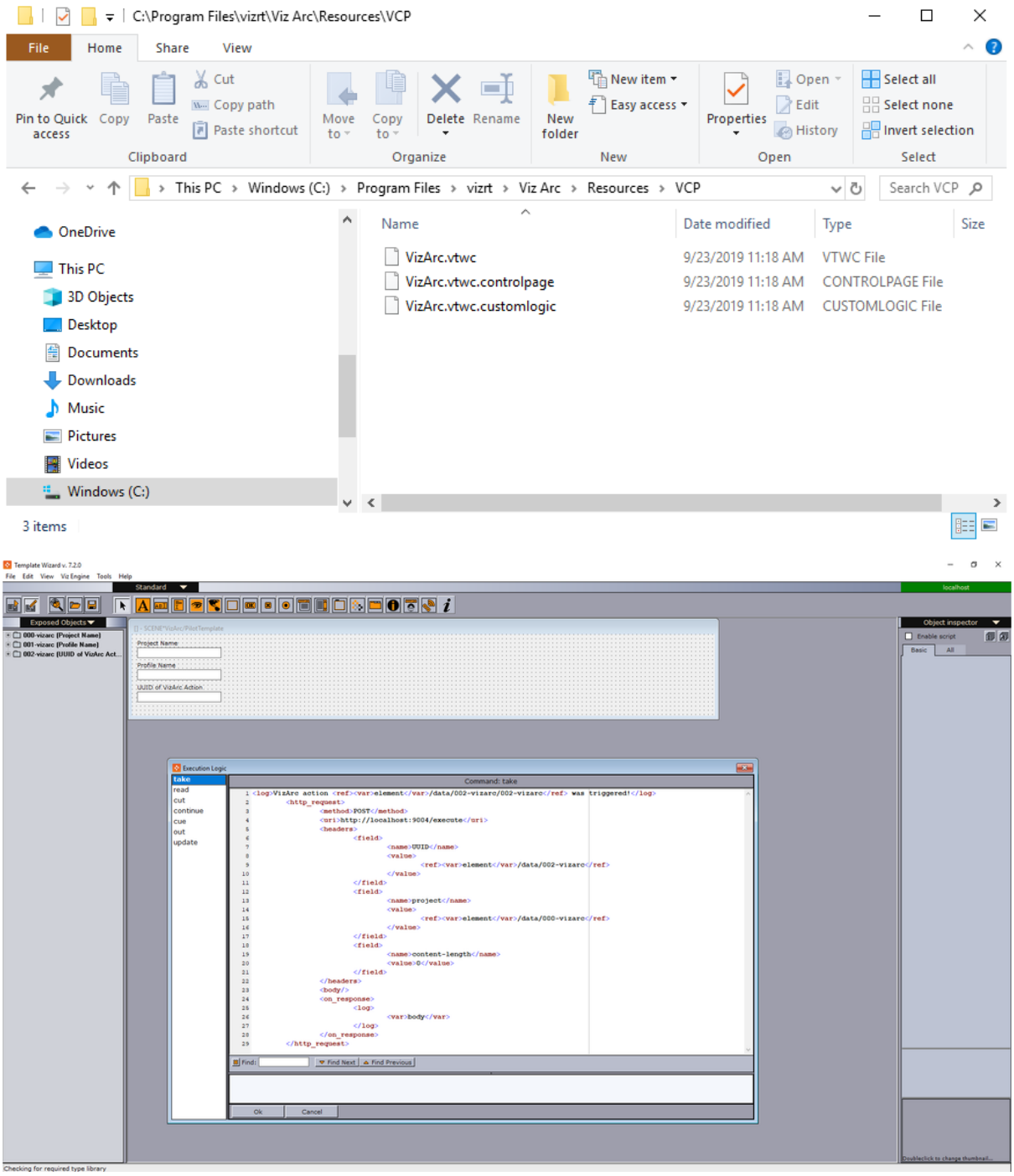
Press the **Remove All** button to remove all actions in the project. You can undo this operation by clicking the **UNDO** icon on the Projects Toolbar.

Save Action as Data to Viz Pilot

Once you have created an action in Viz Arc, you can create a Viz Pilot data element and use it in Viz Director as a normal data element.

1. First, import the Viz Arc Pilot Template using Pilot Template Wizard. The template can be found in the program-data directory (typically *C:\ProgramData\vizrt\VizArc”) under “Resources\VCP*.

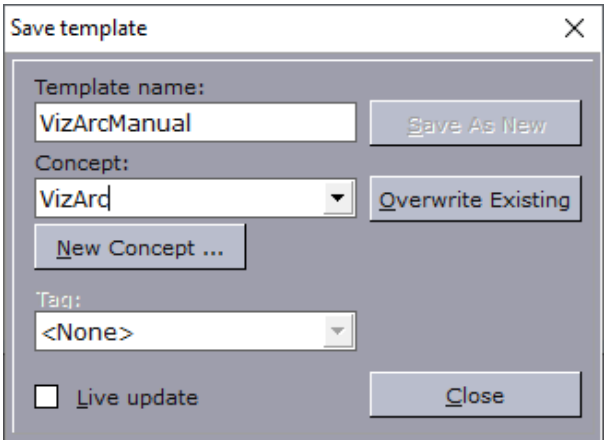
Copyright © 2025 Vizrt



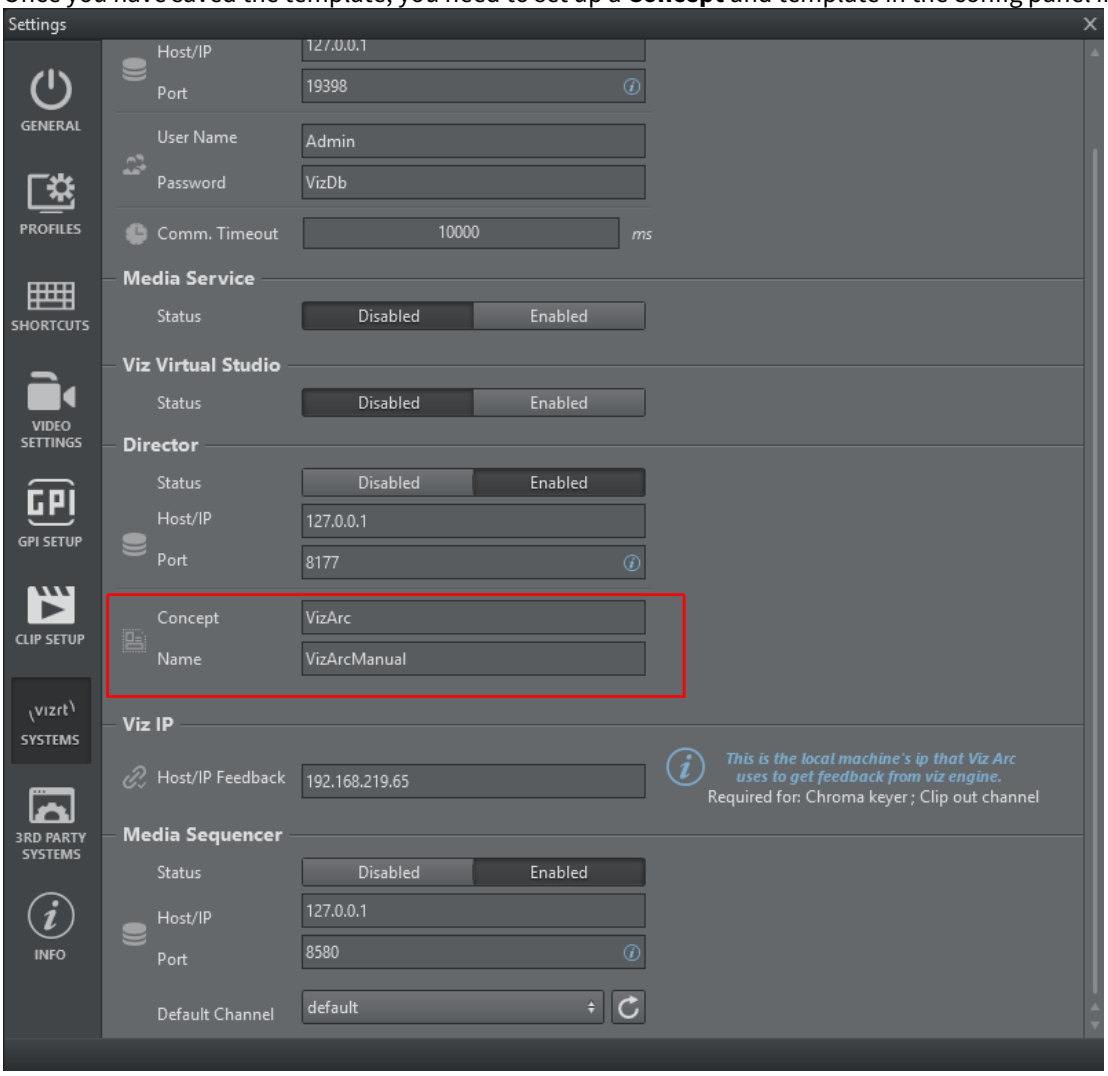
Note: The template has a custom execution logic that performs a REST call when the Viz Pilot data element or template is taken On Air.

2. Create your template and save it.

Copyright © 2025 Vizrt

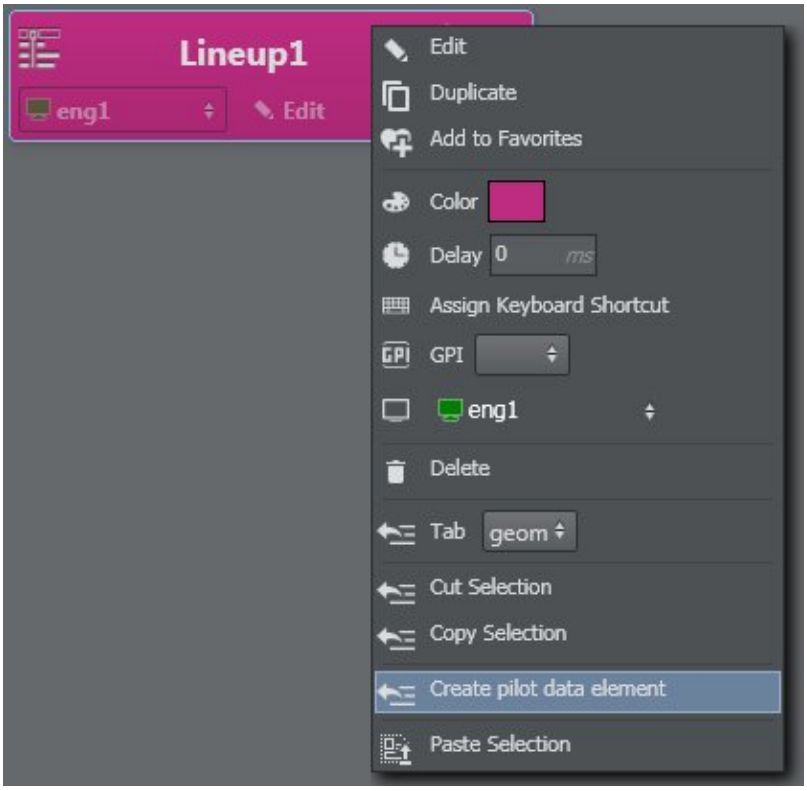


3. Once you have saved the template, you need to set up a **Concept** and template in the config panel in Viz Arc.

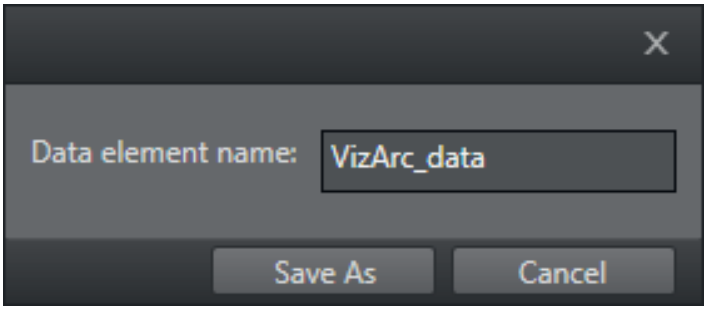


4. Create an action or select an existing action and right-click on it.

Copyright © 2025 Vizrt

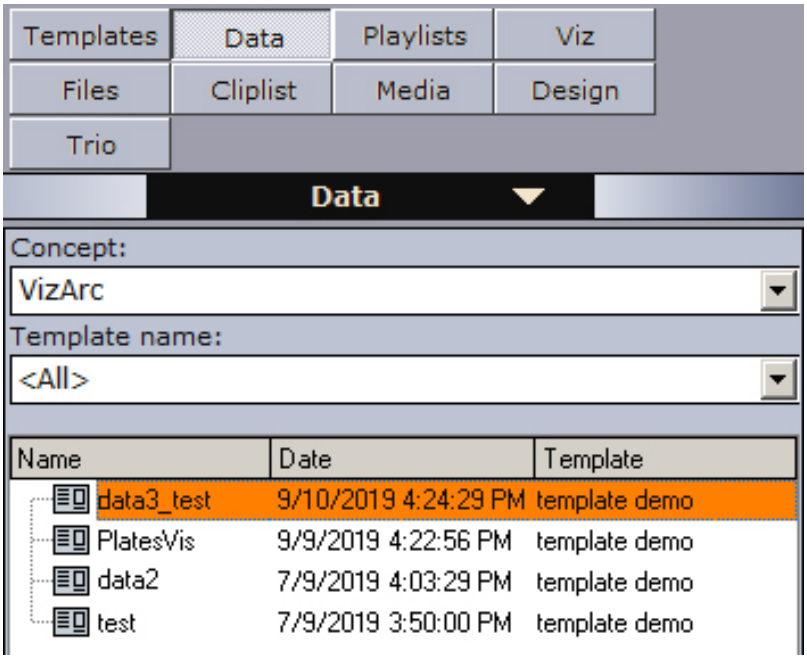


5. Select **Create Pilot data element.** The window below appears. Insert a name for the new Viz Pilot data element.

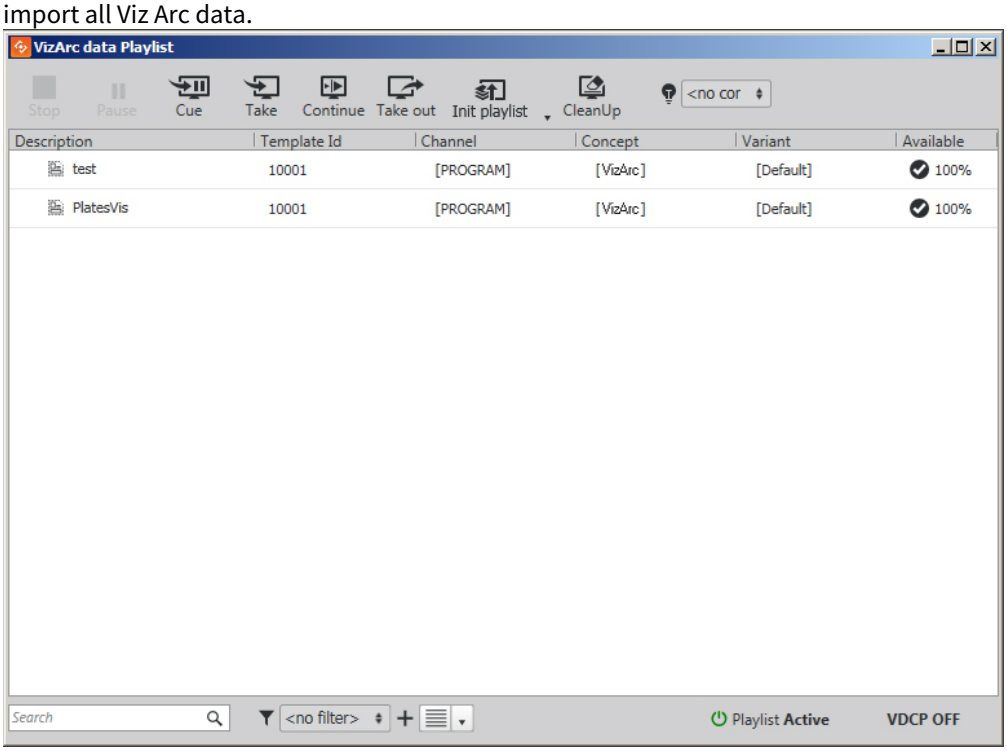


6. Now, switch to Viz Director and you should see your new data in the **Data** section. If you can't see it, right-click on the data list and press **Refresh**.

Copyright © 2025 Vizrt

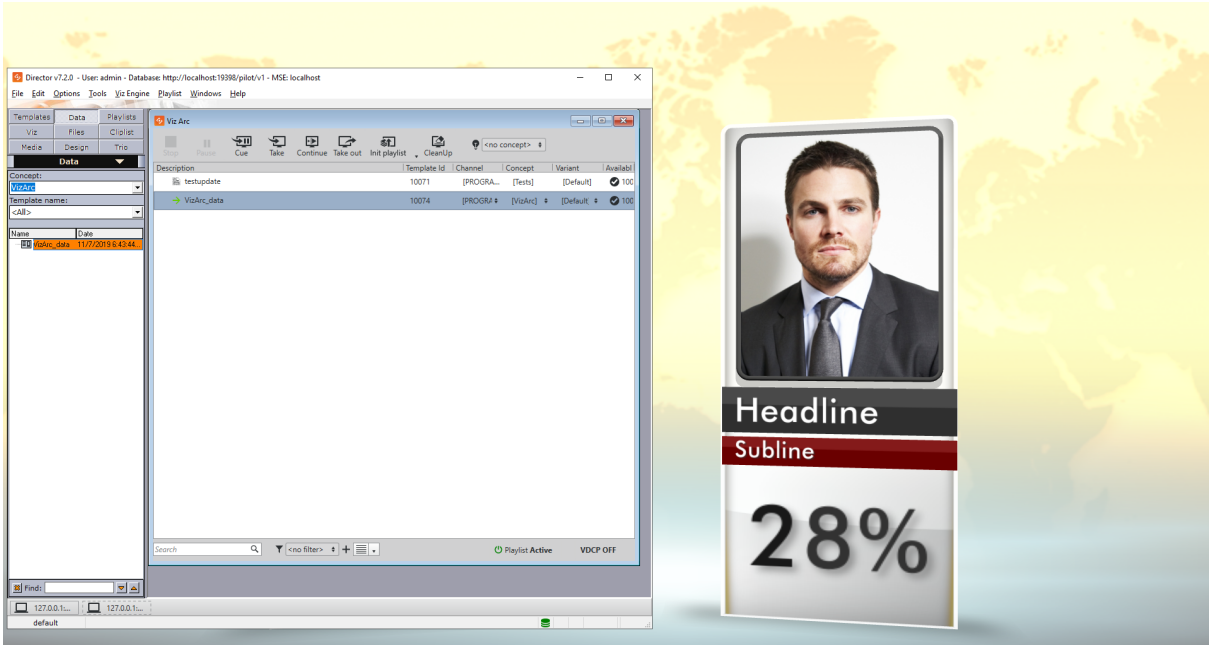


7. Once this data is saved, you can operate Director like normal; for example, you can create a playlist and import all Viz Arc data.



8. Once you have a full playlist, you can go **On Air** with the Director.

Copyright © 2025 Vizrt



See Also

- Supported Action Types
- Viz Pilot User Guide

Copyright © 2025 Vizrt

5.5.2 Set View

Viz Arc lets you work with both outdoor and indoor Augmented Reality events. You can search for a specific place in maps and set up your outdoor set or import an AutoCad project and set up the entire elements environment in the scenography.

> **Note:** Viz Arc uses the metric system.

This section covers the following topics:

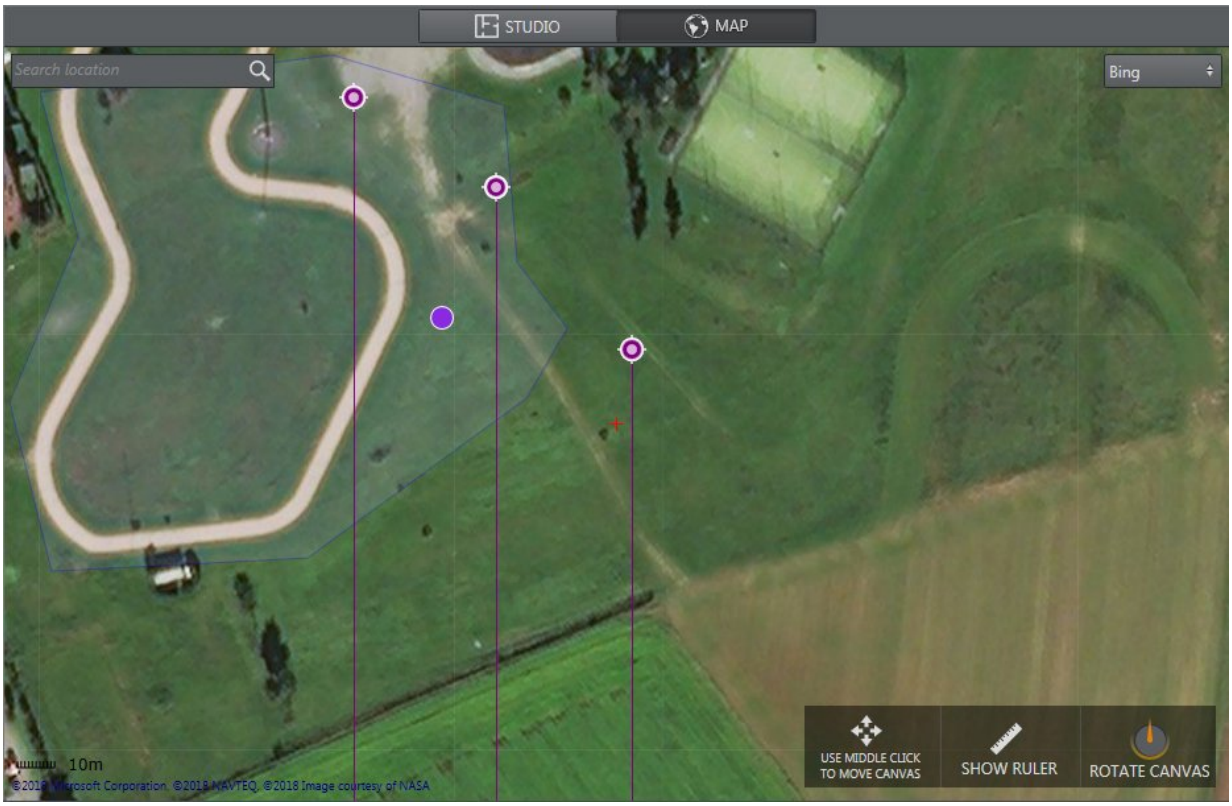
- Map
- Studio
- Working with Virtual Studio
    - To Position an AR Element
    - To Add a Calibration Point
    - To Set up a Camera
    - To Draw a Polygon
    - To Set up a Height Area
    - To Set a Zero Point
- 4 Point Calibration
    - Prerequisites
    - Adding Calibration Points
- Calibration
    - Applying Calibration

Map

Map items can retrieve map data from Bing.

Copyright © 2025 Vizrt

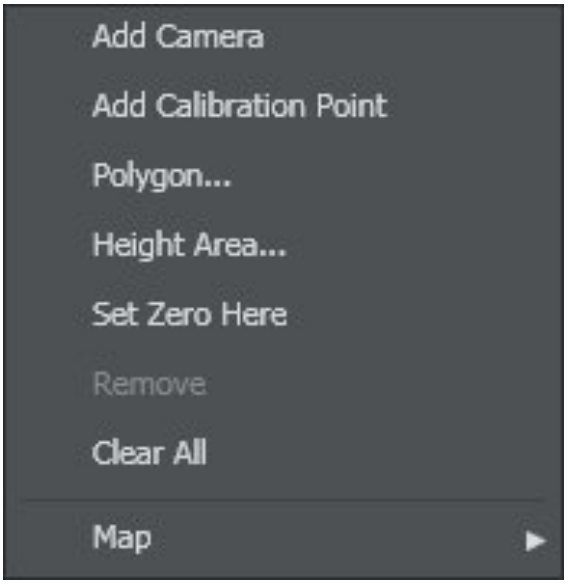
Bing Map



Map is used to set up an external environment. You can search for a place in the search bar and place elements directly info the map area. Map uses XYZ positions OR geolocation and transforms them into Viz Units. The map menu lets you:

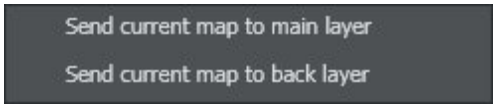
- Zoom in/out with the **mouse wheel**.
- Move the map with the **center mouse button**.
- **Left-click** on an element to move it.
- **Right-click** on an element to remove it.
- **Right-click** on a map:

Copyright © 2025 Vizrt



- Set up a Camera
- Add a Calibration Point
- Draw a Polygon
- Set up a Height Area to intercept the height of the object
- Set a Zero Point
- Clear All elements in map (except for the Zero Point)

By selecting a **Map** item, you can choose to send your current map to the main layer or back layer:



Studio

It's possible to import an AutoCAD project (.dxf) and set up cameras and objects directly into your custom studio using **STUDIO**.

Copyright © 2025 Vizrt

Indoor Map



Studio is used to set up an internal environment. You can import your Studio map project using the **Browse** icon and place elements directly into the plan area. Studio uses Viz Units only.

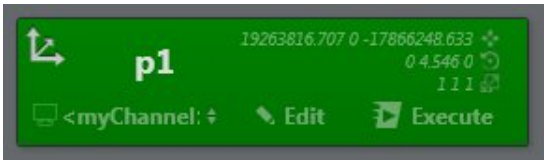
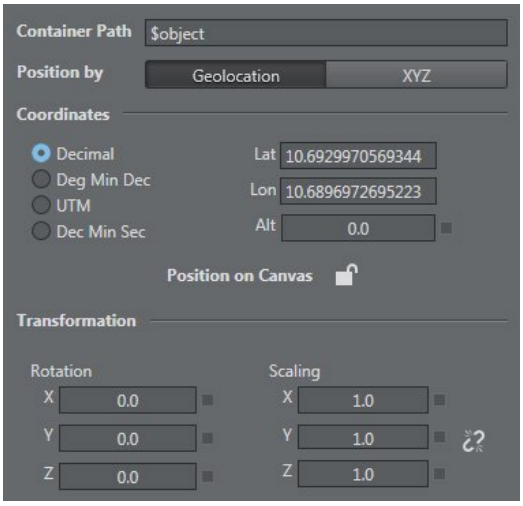
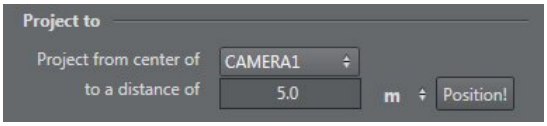
Working with Virtual Studio

This section covers how to set up virtual studio and augmented reality elements in relation to your physical environment.

- To Position an AR Element
- To Add a Calibration Point
- To Set up a Camera
- To Draw a Polygon
- To Set up a Height Area
- To Set a Zero Point
- Prerequisites
- Adding Calibration Points
- Applying Calibration

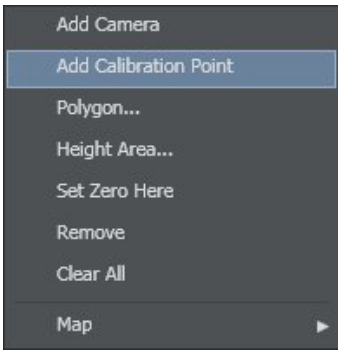
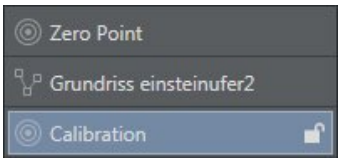
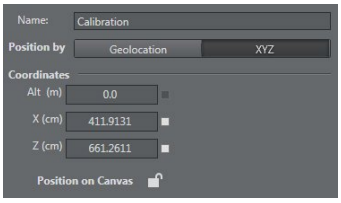
Copyright © 2025 Vizrt

To Position an AR Element

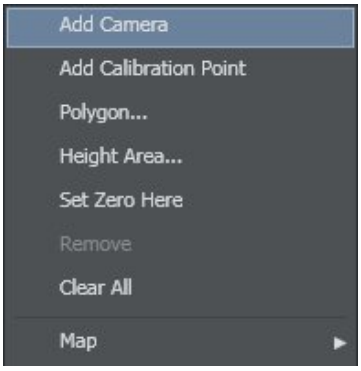
	Drag and drop the desired container's plugin (only position or transformation control channel) into the map area.
	The green icon appears on the map.
	The same elements action that was just created can also be found in the action pane in the main window.
	Selecting the icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	This menu lets you position graphics along the camera center at a given distance.
	This icon lets you lock or unlock an element position on the map canvas.

Copyright © 2025 Vizrt

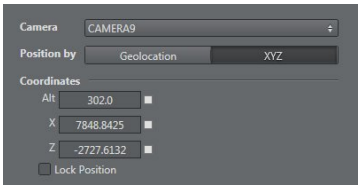
To Add a Calibration Point

	Right-click on the map OR in the studio area and select **Add Calibration Point** in the context menu.
	The target icon appears on the map.
	The item is also added to the element list on the right of the map/ studio area.
	Selecting an icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	It's possible to lock or unlock a camera position on the map canvas using this command.

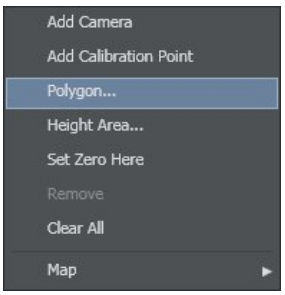
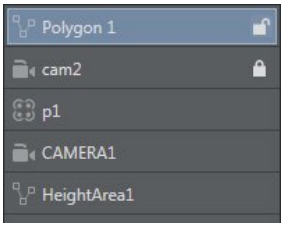
To Set up a Camera

	Right-click on the map OR in the studio area and select **Add Camera** in the context menu.

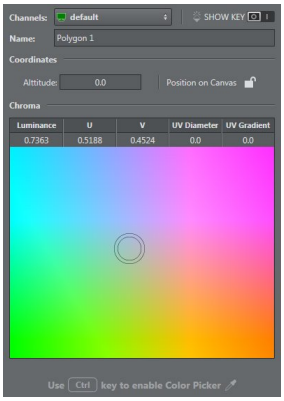
Copyright © 2025 Vizrt

	The camera icon appears on the map.
	The item is also added to the element list on the right of the map/ studio area.
	Selecting an icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	It's possible to lock or unlock a camera position on the map canvas using this command.

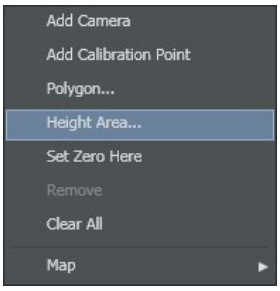
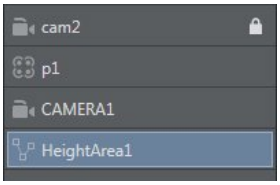
To Draw a Polygon

	Right-click on the map OR in the studio area and select **Polygon...** in the context menu.
	The cursor transforms into a pen and you can start to draw the polygon on the map. Mark as many vertices as you need.
	The item is also added to the element list on the right of the map/studio area.

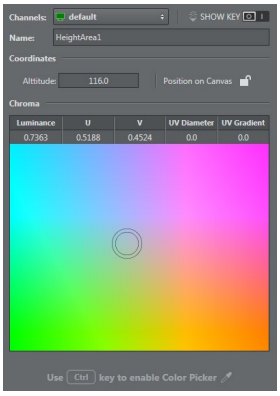
Copyright © 2025 Vizrt

	Selecting an icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	It's possible to lock or unlock a camera position on the map canvas using this command.

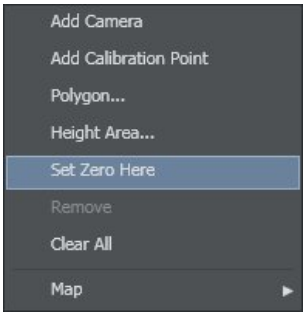
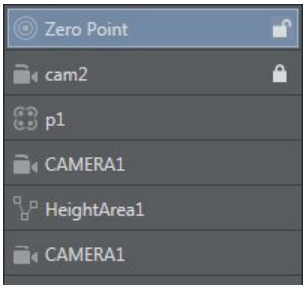
To Set up a Height Area

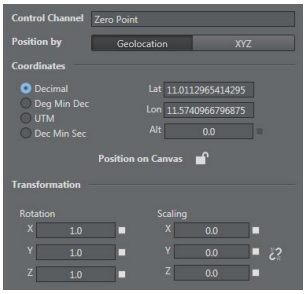
	Right-click on the map OR in the studio area and select **Height Area...** in the context menu.
	Click once to set a point and click again to set the height. This icon appears on the map.
	The item is also added to the element list on the right of the map/studio area.

Copyright © 2025 Vizrt

	Selecting an icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	It's also possible to lock or unlock the height area position on the map canvas.

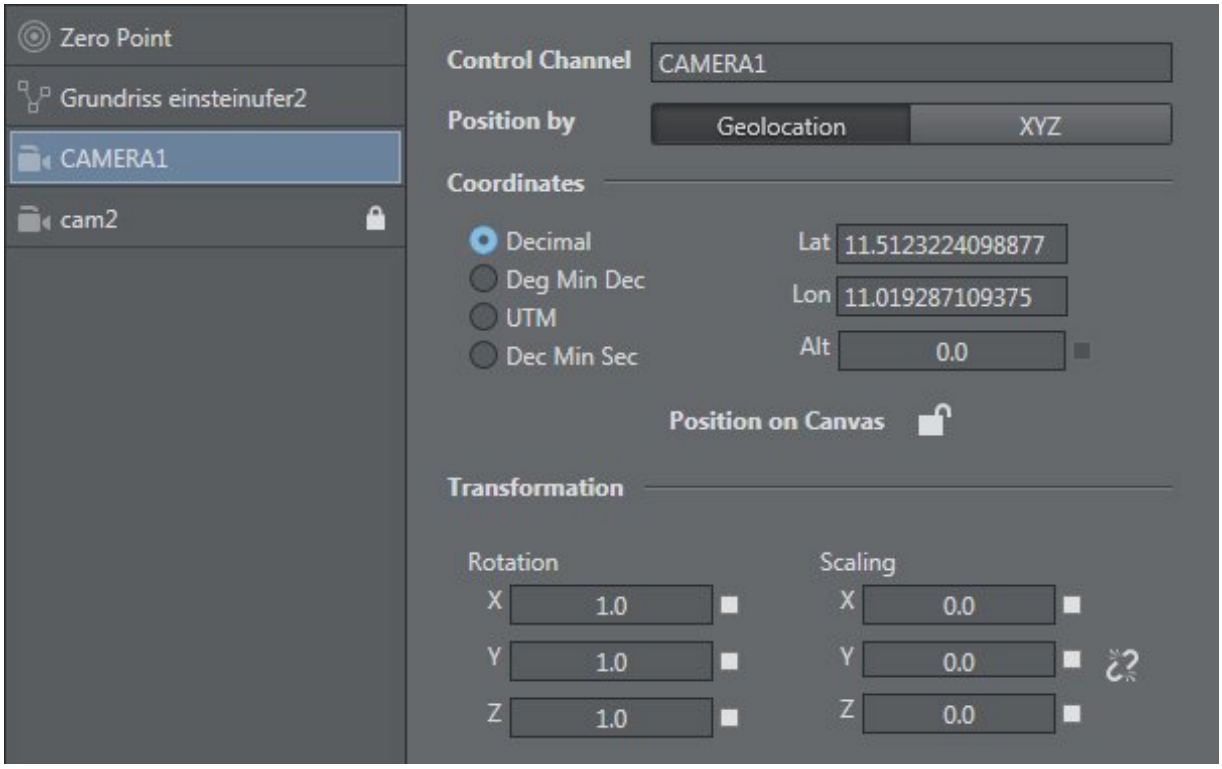
To Set a Zero Point

	Right-click on the map OR in the studio area and select **Set Zero Here** in the context menu.
	This icon appears on the map.
	The item is also added to the element list on the right of the map/ studio area.

	Selecting an icon or element in the list makes the corresponding config section appear on the right side of the SET window.
	It’s also possible to lock or unlock a zero point position on the map canvas.

Once an element (like a camera) has been added, it's possible to edit the parameters:

- Position – choose between
  - **geolocalization**: latitude, longitude, altitude, and type: **Decimal**, **Deg Min Dec**, **UTM** , **Dec Min Sec**
  - **XYZ**
- Rotation
- Scaling

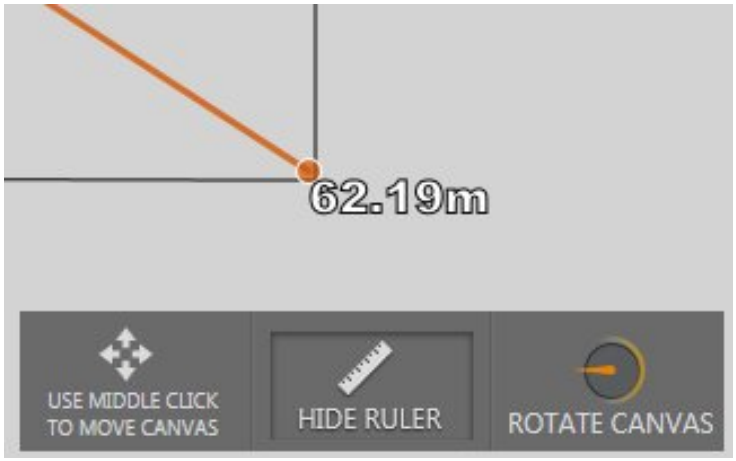


You can toggle **show ruler** to make small adjustments to element positions on the map. **Rotate canvas** by holding down the left cursor button to the desired map position.



Copyright © 2025 Vizrt

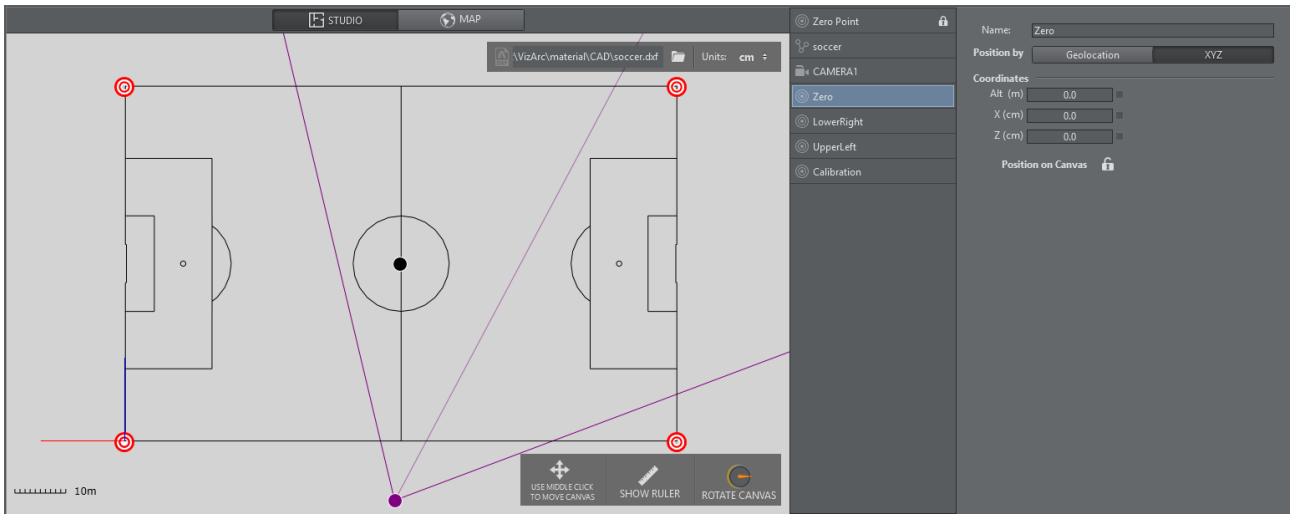
Tip: The Ruler tool is useful since it allows you to measure the distances on your area in Viz Arc in real time.



Note: It's possible to delete all elements in a map except the Zero Point.

4 Point Calibration

4 Point Calibration can be used to calculate the position and orientation of a (typically mechanically) tracked camera. Using this type of calibration requires knowledge of at least four physical points in the domain (for example, the corners of a soccer pitch or well-known points on a horse race track).



Map View showing a soccer pitch and 4 calibration points.

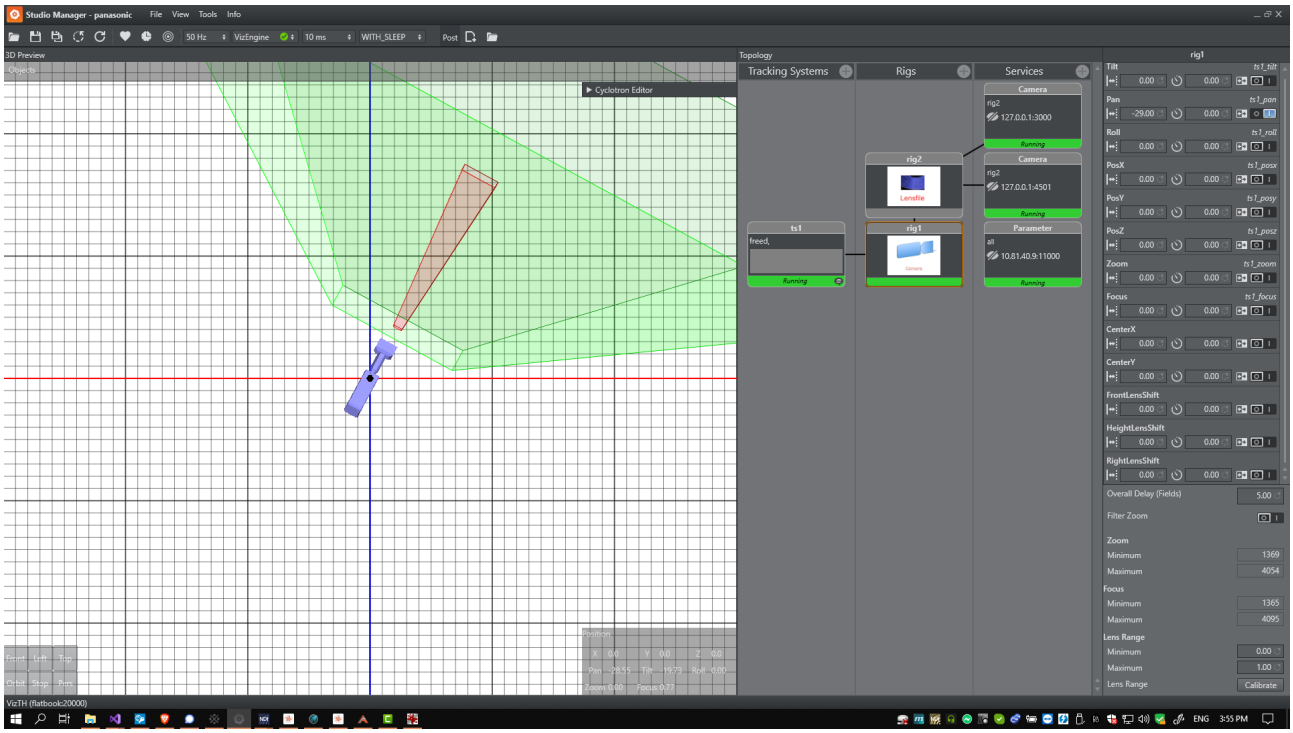
Prerequisites

- The calibration points must all be on the XZ plane (same height).
- The camera must be on a fixed position and can't be moved in space.
- At least four points need to be measured/known on the tracking domain.
- The points must define a surface and not just a one dimensional line.

Copyright © 2025 Vizrt

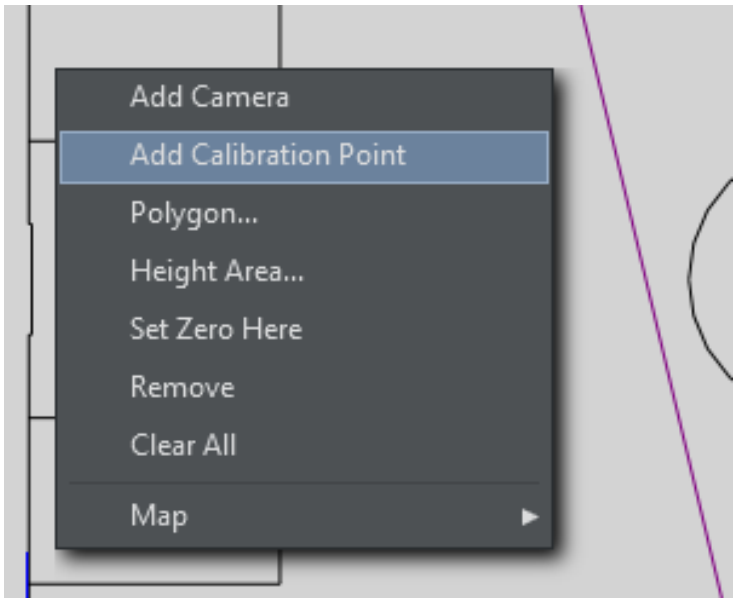
- Make sure the tracking hub sends the tracking information to Viz Arc and make sure it includes the camera lens rig containing a valid calibration.

Note: Do NOT use a parent transformation rig on top for 4 Point calibration. The translation and rotation are handled inside the root node of the Viz Artist scene.

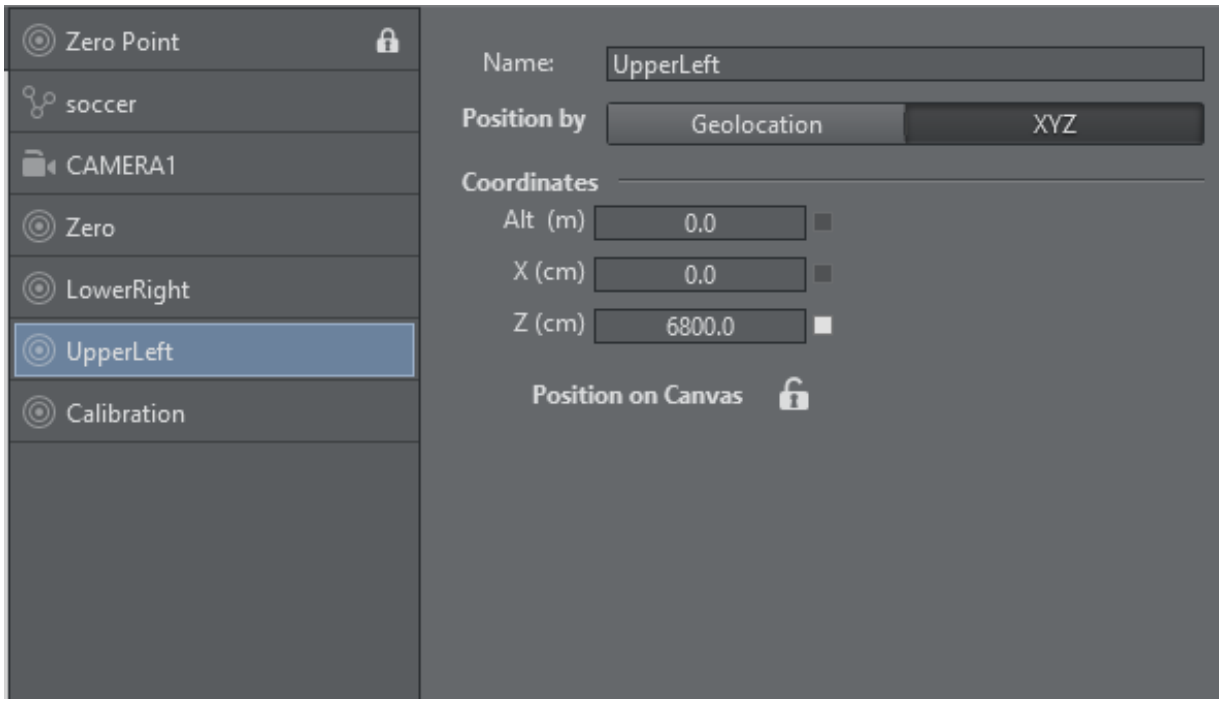


Adding Calibration Points

Add calibration points from the map's context menu:



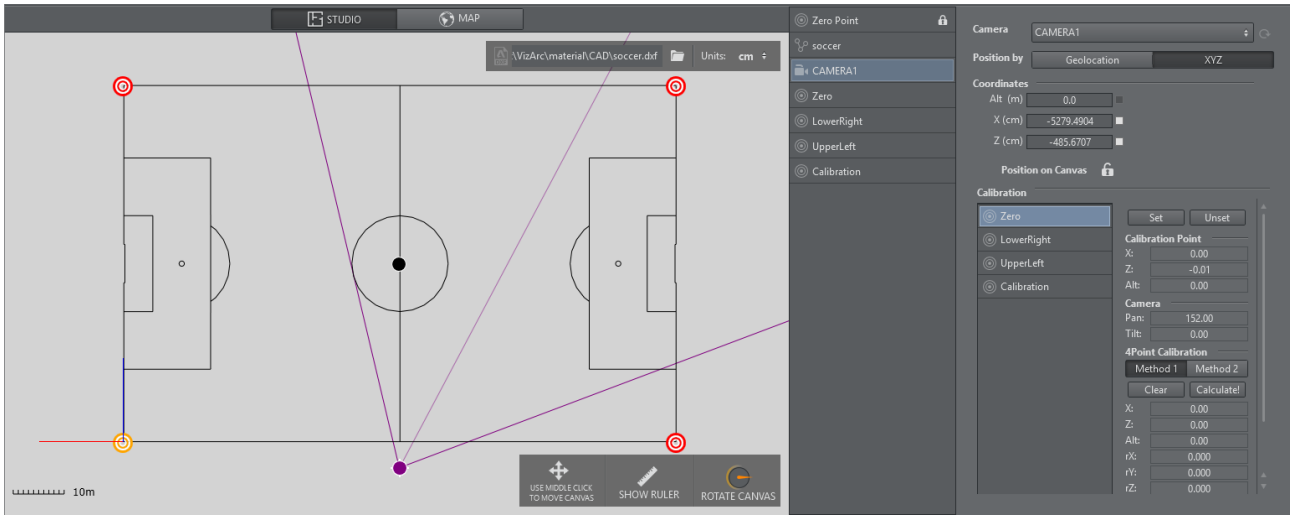
Select the calibration point you just added and move it on the map, or enter its coordinates manually:



Once all calibration points have been added and positioned, the actual calibration process may start.

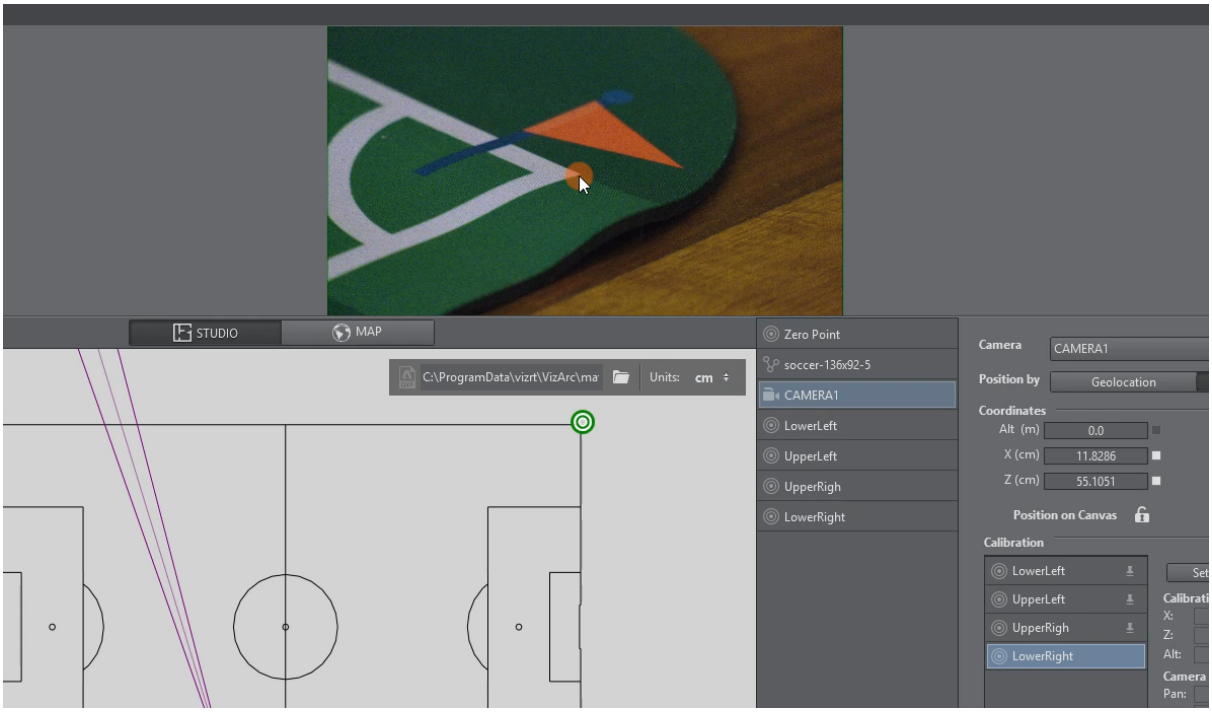
Calibration

Select the camera to be calibrated.



1. On the camera user interface select the marker towards which you want to point the camera. The calibration point is colored **yellow** on the map canvas.
2. The physical camera should now fully zoom into the marker and try to center it as best as possible.
3. Once the marker is centered, click the **Set** button. If your preview has been set up so that the camera is visible and you have a valid lens calibration, you can also click the position in the view instead of using the **Set** button.

Copyright © 2025 Vizrt

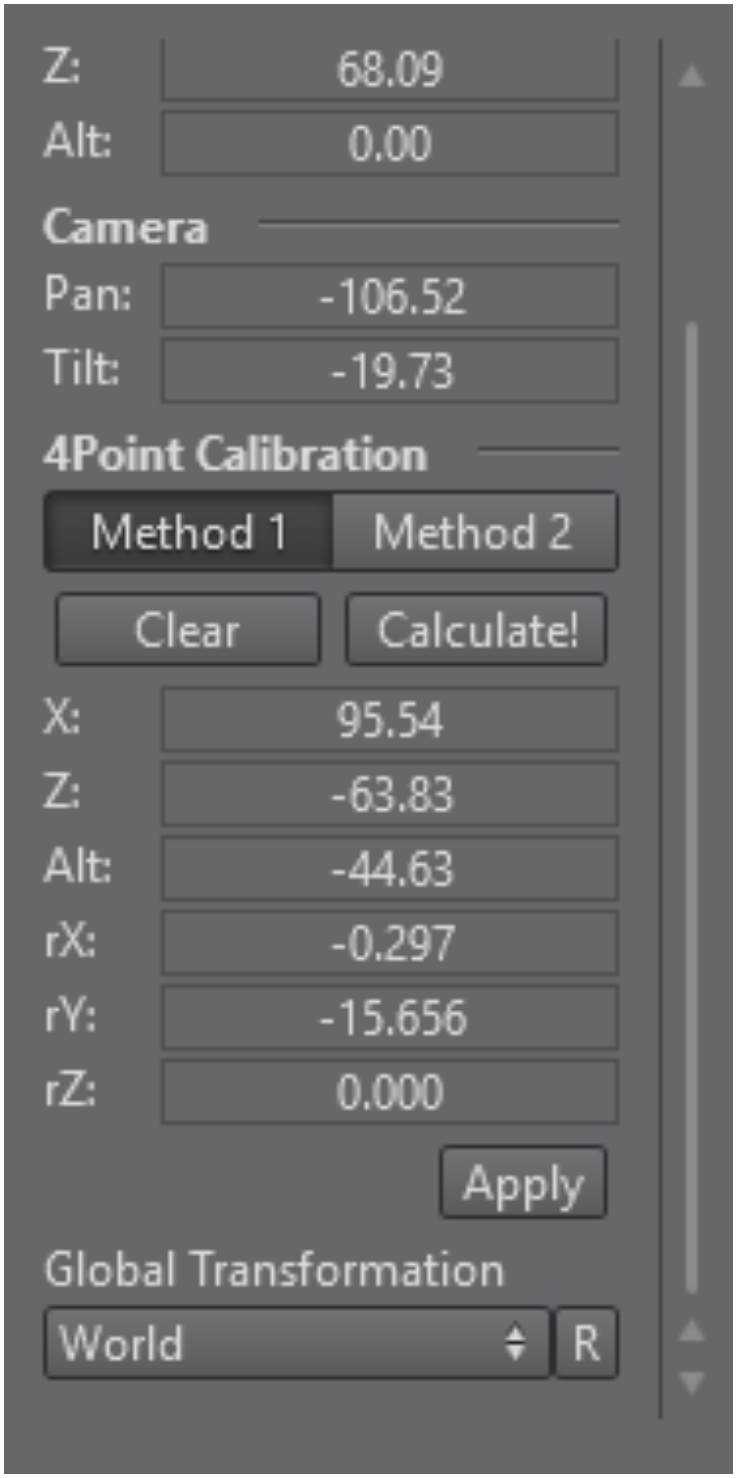


4. Continue this procedure for all or at least 4 calibration points.

Applying Calibration

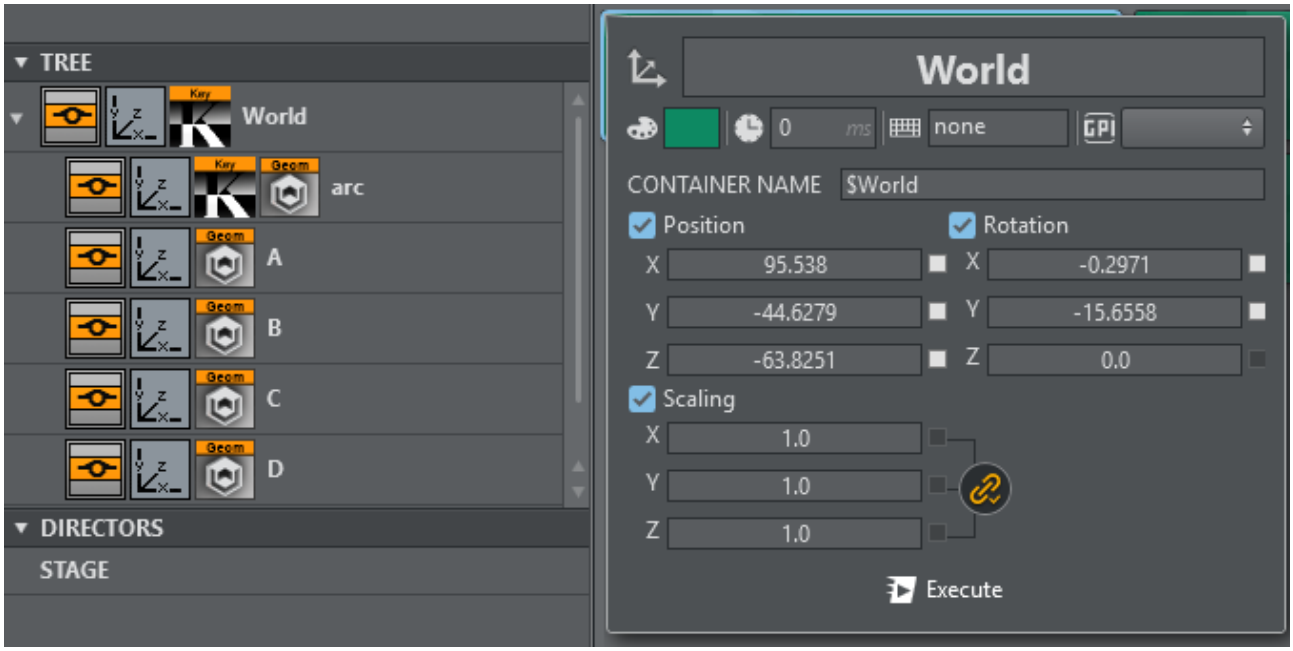
1. Once at least four points have been set, click the **Calculate!** button to calculate the presumed position and orientation of the camera. You can choose between **Method 1** and **Method 2**, hit **Calculate!** after switching algorithm. Generally, Method 2 is more accurate but needs more and more accurate calibration points. Try both and pick the one that gives you better visual results.
2. Under the **Global Transformation** dropdown, select the transformation action holding the world transformation of the main scene (typically the root node of the AR/VR scene).
3. Click **Apply** to apply the calculated transformation and rotation.

Copyright © 2025 Vizrt

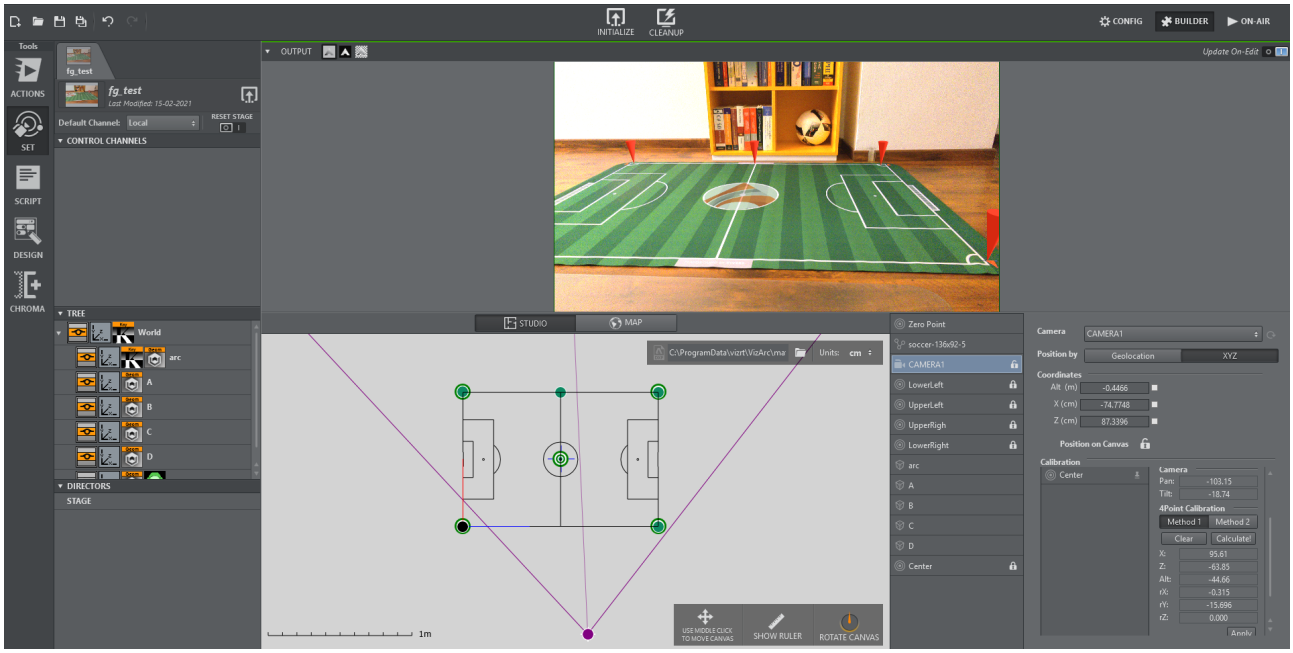


Use the **R** button to refresh the list of available transformation actions.

Copyright © 2025 Vizrt



Typically the transformation to be used is the root node of the scene.

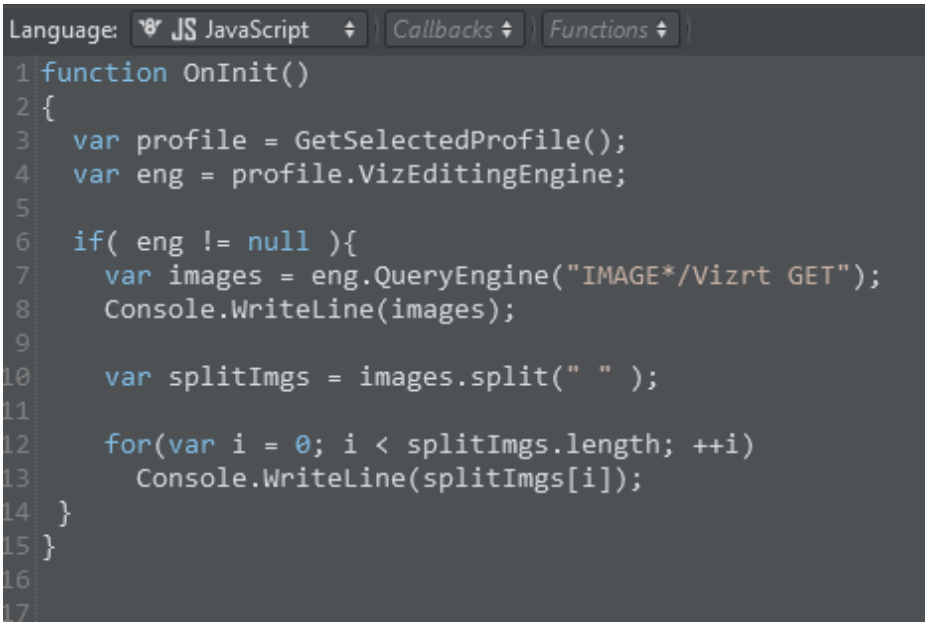


Once calibrated you should be able to position AR graphics on your pitch.

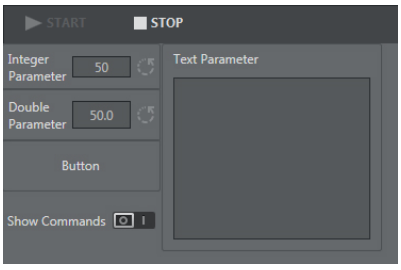
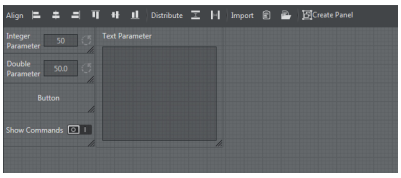
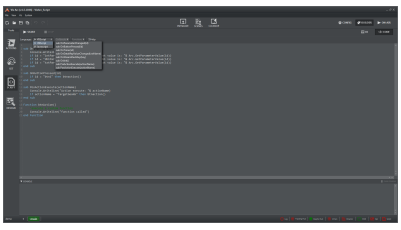
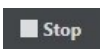
Copyright © 2025 Vizrt

5.5.3 Script View

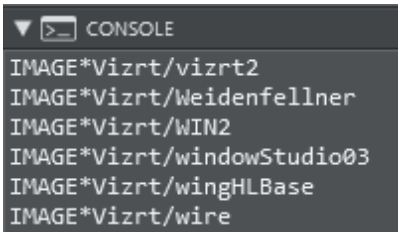
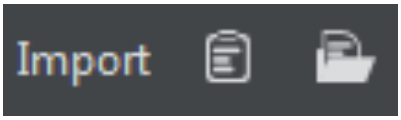
In Script View, you can write your own custom script in JavaScript language through Google's **V8** or Microsoft's **JScript** (ECMAScript3) or in **VBScript** language, as in the following example:

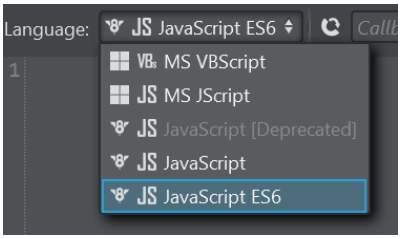


It's possible to create custom forms and components, such as text boxes and buttons.

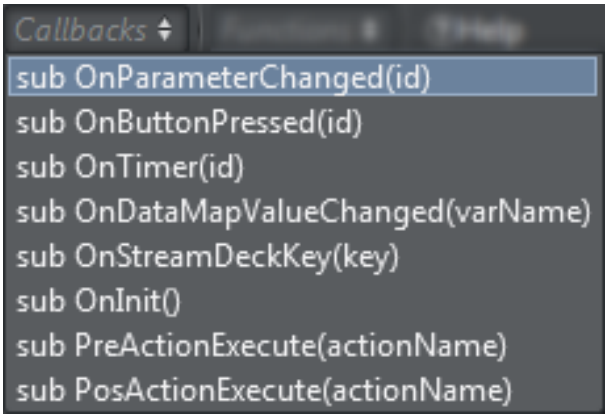
	To run the script, select the **Start** button  on the top left of the window.
	Form Design can be edited by selecting the **UI** button UNKNOWN ATTACHMENT. Every element can be selected and moved, aligned and distributed on the main form. To go back to code editing, select the **CODE** button.
	To edit a script, press the **Stop** button  on the top of the script main window.

Copyright © 2025 Vizrt

	Console logs and debugs are displayed in the **CONSOLE** pane.
	Import code internally from the clipboard or an external text file in the script pane.

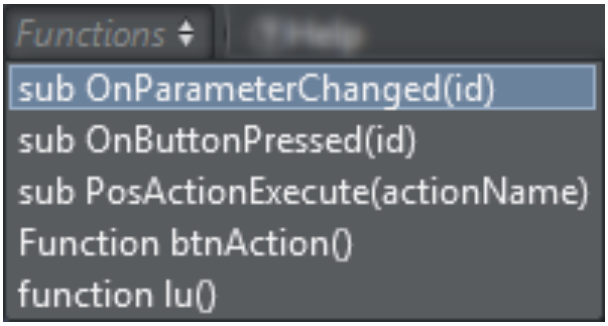
	Use the **Language** menu to select a scripting language.
	• MS VBScript Microsoft Visual Basic Scripting language)
	• MS JScript Microsoft implementation of the ECMA 262 JavaScript
	• JavaScript Google's open source high-performance V8 JavaScript
	  • Using CommonJS syntax to import modules (using **require**).
	• JavaScript ES6 Google's open source high-performance V8 JavaScript
	  • Using ES Modules syntax to import modules (using **import**)
	It is recommended to use **JavaScript ES6** language when possible.

In edit mode, **Script Callbacks** can be selected from the list and added:



You can locate a custom function by selecting it from the **Functions** list:

Copyright © 2025 Vizrt



See Also

- Scripting Classes

Control Object Classes

After having accessed the action holding the list of ControlObjects thought the **GetAction** method, the single ControlObjects can be retreived using the global method

- BaseControlObject **GetControlObject**(BaseAction action, string ControlObjectID)

Most Control Object types have the following generic properties:

- **Text** (String)
  - This property adapts to all objects (execute string)
  - `IntControl.Text = "5"`
  - `ImageControl.Text = "IMAGE*FolderA/SubfolderB/ImageName"`
- **ID** (String)
  - Returns ObjectID
- **Description**
  - Returns the object description

Each Control Object type has specific properties:

- Control Container
- Control Image
- Control Material
- Control Omo
- Control Text
- Control List
  - Single Cells Properties
- Control Integer
- Control Double
- Control Boolean

Control Container

Properties:

- Visibility
- Position
  - posX (double)
  - posY (double)
  - posZ(double)
- Rotation
  - rotX (double)
  - rotY (double)
  - rotZ (double)
- Scaling
  - scaX (double)
  - scaY (double)
  - scaZ (double)

Copyright © 2025 Vizrt

This type doesn't have the Text property.

Control Image

Properties:

- Path (string)
- Position
    - posX (double)
    - posY (double)
- Scaling
    - scaX (double)
    - scaY (double)

Control Material

Properties:

- Path (string)

Control Omo

Properties:

- Value (integer)

Control Text

Properties:

- Value (string)

Control List

Example:

```
sub OnInit()
    'declare variables
    dim objAction, table, cell
    dim output1, output2, output3
    'get table obj action
    objAction = arc.GetAction("object")
    table = arc.GetControlObject (objAction, "controlObj_ID")
    Console.WriteLine("Table name: " & table.Text)
    'set values in single cells inside the table
    table(0,0).value = false
    table(0,1).value = 5
    table(2,5).x = 12
    table(3,6).value = "IMAGE*/Default/GER"
    'assign values to a variable and show in debug console
    output1 = table(0,0).Text
```

Copyright © 2025 Vizrt

```
    output2 = table(0,1).Text
    output3 = table(0,2).Text
    Console.WriteLine("cell - " & output1 & " | " & output2 & " | " & output3)
end sub
```

Properties:

- Accessor
  - **table[int row, int col]:** returns a cell
  - **nbcolumns (integer):** number of columns
  - **nbrows (integer):** numbers of rows

Single Cells Properties

Cell	Type	Additional Information	Example
BaseCell	Text (string)	Sets or gets value as string. Common to every cell type.	`table(0,5).x= 12 (intCell)` `table(0,5).text = "12" (intCell)`
BoolCell	Value (boolean)		`table(0,5).active= true`
DoubleCell	Value (double)		table(0,5).x= 12.8
DupletCell	X (double) Y (double)		table(0,5).text = "0.55 0.2"
GeomCell	Value (string)		table(0,5).value = "GEOM*/folder/geometry"
ImageCell	Value (string)		table(0,5).value = "IMAGE*/folder/image"
IntCell	Value (integer)		
MaterialCell	Value (string)		table(0,5).value = "MATERIAL*/folder/material"
TextCell	Value (string)		
TripletCell	X (double) Y (double) Z (double)		table(0,5).text = "0.55 0.23 1.23"

Copyright © 2025 Vizrt

Control Integer

Properties:

- Value (integer)

Control Double

Properties:

- Value (double)

Control Boolean

Properties:

- Value (boolean)

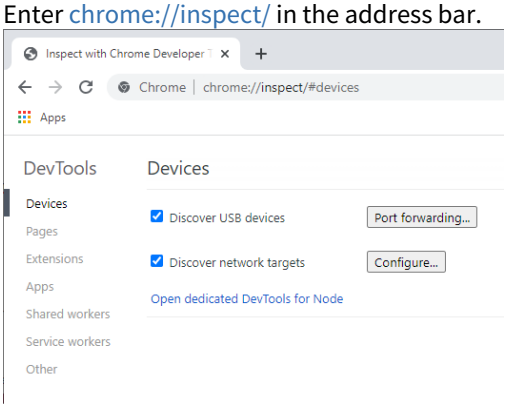
Copyright © 2025 Vizrt

Debugging

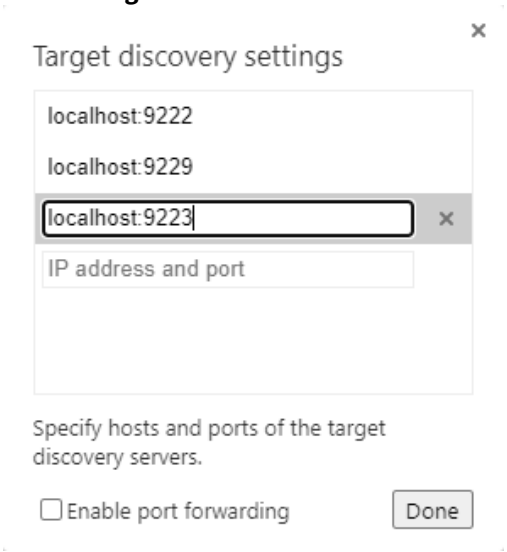
DevTools

You can use any chrome based web browser (for example, Google Chrome or Microsoft Edge) to step through Viz Arc scripts.

1. Open your browser.
2. Enter chrome://inspect/ in the address bar.

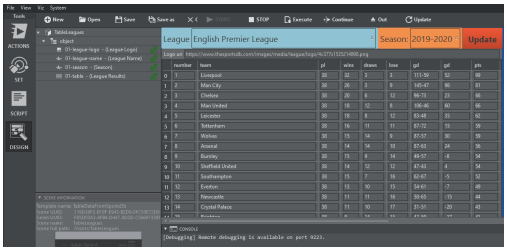


3. Click **Configure...**

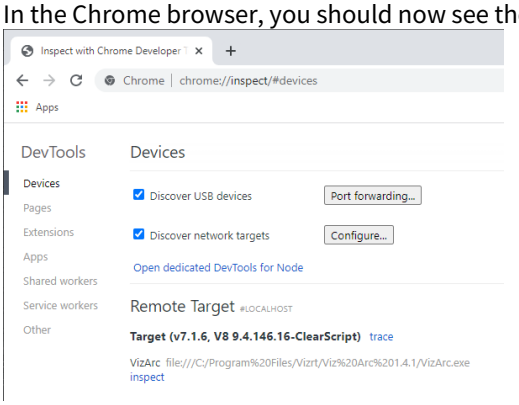


   Add the host name where Viz Arc is running and specify the debug script port (by default, port `9222` for the global script and port `9223` for the template builder script). Confirm by clicking the **Done** button.
4. Open a template in Viz Arc's template builder and start it.

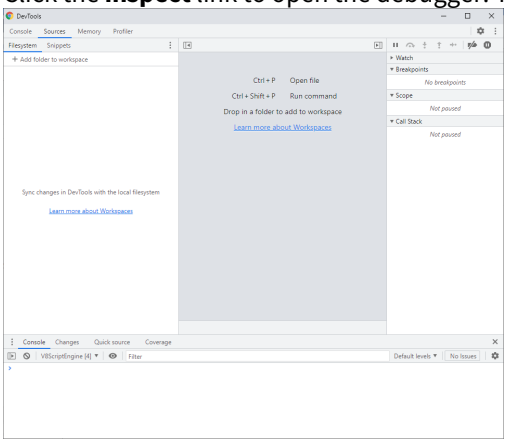
Copyright © 2025 Vizrt



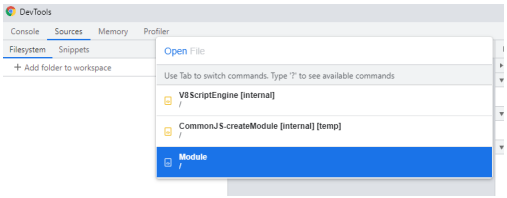
5. In the Chrome browser, you should now see the Viz Arc script available for debugging.



6. Click the **inspect** link to open the debugger. The first time you open the debugger it does not show any code.

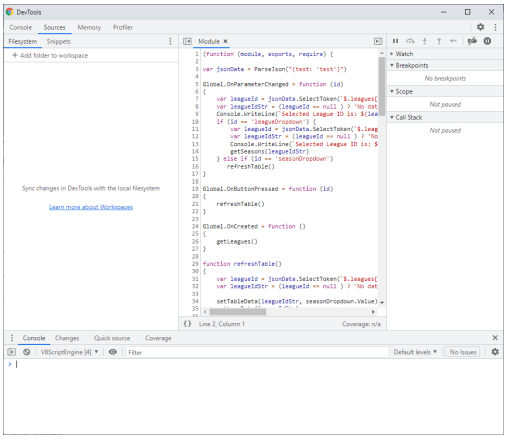


Hit **CTRL + P** and select *Module*.

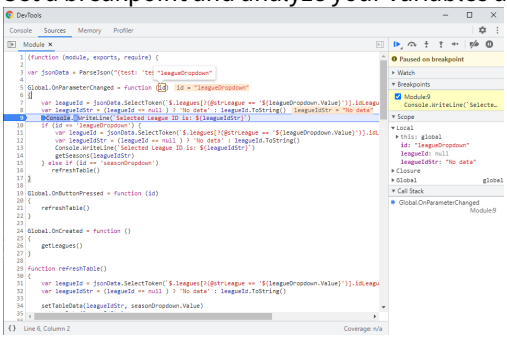


You'll now be able to set breakpoints and see the code.

Copyright © 2025 Vizrt



7. Set a breakpoint and analyze your variables and code.



Visual Studio Code

You can debug scripts with Visual Studio Code when using V8 JavaScript in the global script or in any scripted template.

1. Install and launch Visual Studio Code.
2. Set up one or more Viz Arc V8 debug configurations:
    a. Click **File** > **Preferences** > **Settings** to open your user settings.
    b. Locate or search for the **Launch** configuration and click **Edit in settings.json**.
    c. Add the following section to the file:

```
{
 "debug.javascript.usePreview": false,
 "launch": {
    "version": "1.2.0",
    "configurations": [
        {
            "name": "Attach to Viz Arc Global Script on port 9222",
            "type": "node",
            "request": "attach",
            "protocol": "inspector",
            "address": "localhost",
            "port": 9222
        },
        {
            "name": "Attach to Viz Arc Template Scrip on port 9223",
```

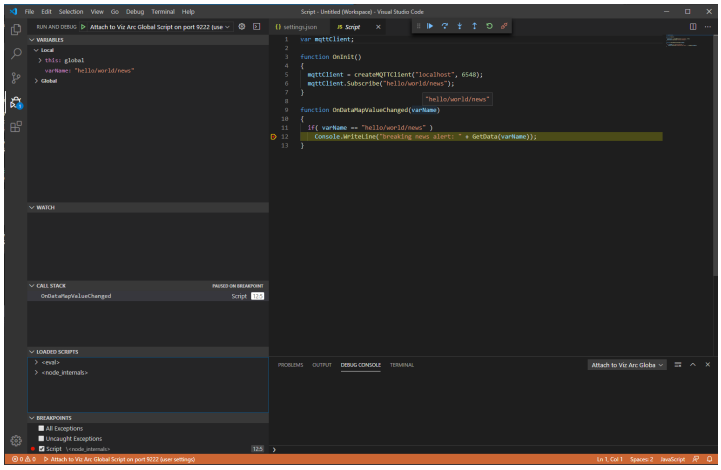
Copyright © 2025 Vizrt

```
            "type": "node",
            "request": "attach",
            "protocol": "inspector",
            "address": "localhost",
            "port": 9223
        }
    ]
  }
}
```

d. You can specify additional configurations for different hosts, port numbers, and other options. See Node.js debugging in VS Code for more information.
e. Click **File > Save**.

3. If you'd like to debug your application remotely, you must also make sure that your firewall allows incoming connections to your TCP port.
4. Attach the Visual Studio Code debugger to your application:
   a. Click **View > Debug** to bring up the Debug view.
   b. Select the appropriate debug configuration at the top of the Debug Side Bar.
   c. Click **Debug > Start Debugging**.

Note: There are two different ports in use: One for the global script (default `9222`) and one for template scripts (default `9223`). Template scripts can be debugged only when running in the designer. The ports can be configured in the global configuration settings.



The screenshot above shows a global script being stopped at a breakpoint.

See Also

- Node.js debugging in VS Code

Copyright © 2025 Vizrt

Profiles Classes

This section contains a list of properties and functions grouped by type that are useful for communicating with Profile, Channel, and Engine (Viz Engine and Unreal Engine).

- Scripting Profile
- Scripting Channel
- Scripting Engine

Scripting Profile

- string **Name** [Get]
    - Returns the profile's name
- int **NumChannels** [Get]
    - Returns the number of channels
- ScriptingChannel **VizEditingEngine** [Get]
    - Returns the configured Viz Editing Engine of the profile
- ScriptingChannel **UnrealEditingEngine** [Get]
    - Returns the configured Unreal Editing Engine of the profile
- ScriptingChannel **VizProgramChannel**[Get]
    - Returns the configured Viz program Engine of the profile
- ScriptingChannel **UnrealProgramChannel**[Get]
    - Returns the configured Unreal program Engine of the profile
- ScriptingChannel **VizPreviewChannel**[Get]
    - Returns the configured Viz preview Engine of the profile
- ScriptingChannel **UnrealPreviewChannel**[Get]
    - Returns the configured Unreal preview Engine of the profile
- ScriptingChannel **Accessor** [int index] [Get]
    - Returns the *index*-indexed Scripting Channel
- ScriptingChannel **GetChannel** (int index)
    - Returns the *index*-indexed Scripting Channel
- ScriptingChannel **GetChannel** (string channelName)
    - Returns the first channel found with name *channelName*

Scripting Channel

- string **Name** [Get]
    - Returns the channel's name
- int **NumChannels** [Get]
    - Returns numbers of Engines in the channel
- ScriptingChannel **Accessor** [int index] [Get]
    - Returns the *index*-indexed Scripting Engine Class
- void **SendSingleCommand** (string command)
    - Sends the command to all the Engines in the channel
- void **SendCommands** (string[] commands)
    - Sends a list of commands to all the Engines in the channel

Copyright © 2025 Vizrt

- ScriptingEngine **GetEngineByName** (string name)
    - Return the first Engine found with name

Scripting Engine

- void **SendSingleCommand** (string command)
    - Sends the command to the Engine
- void **SendCommands** (string[] commands)
    - Sends a list of commands to the Engine
- string **QueryEngine** (string command, int timeout = -1)
    - Queries the Engine with command
      If timeout is specified and larger than 0, the method times out after *timeout* milliseconds if it does not receive an answer within that time
- Task<string> **QueryEngineAsync** (string command, int timeout = -1)
    - Queries the Engine with command asynchronously
      If timeout is specified and larger than 0, the method times out after *timeout* milliseconds if it does not receive an answer within that time
- string **GetFlowicsOutput**()
    - Returns the Flowics live output URL associated to the API token of this engine. Return null if it's not a Flowics output engine.

Copyright © 2025 Vizrt

Scripting Classes

This section covers the following topics:

- General
- Action
- Playlist
- Control Object
- MIDI
- Art-Net DMX
- MQTT
- Object Tracker
- Viz Arena
- Parameter
- Channel
- Viz Engine/Unreal Engine Communication
    - Viz Engine Communication
- Tracking Hub Command
- SMM Handling
- GPI
- Viz Pilot
- Timer
- setTimeout
- StreamDeck
- Graphic Hub REST
- DataMap
- NDI
- File Handling
- Logging
- JSON
- Excel
- Callbacks
- Exposed Objects
    - Console
    - MessageBox
    - XmlDocument
    - XMLHttpRequest
    - xlAppType
    - FSO
- xHost
- Performance
- SQL Sample
- SQLite Sample
- TcpSend
- HtmlAgility Example
- Main Script-only

Copyright © 2025 Vizrt

        - Canvas Tabs Handling
        - Action Template Handling
        - Callbacks
    - Template Script-only
        - Action/Designer Handling
        - Control Object Handling
        - Template Channels Handling
        - ScriptingChannel
        - Template Scene Handling
        - Template Action Configuration
        - Callbacks
    - Common Callbacks
    - Parameters
        - Base Parameters Functionality
        - Layout
        - Dialogs
        - Directory
        - File
        - Asset
        - WebView
        - Bool
        - Button
        - Toggle Button
        - Double / Double Slider
        - Dropdown / Radio
        - Int / Int Slider
        - MultiText / Text
        - Triplet
        - Table
    - Unreal
    - Video
    - Using async/await
        - Using try/catch

General

Viz Arc's scripting has many classes and types that are exposed and accessible via code. The script's main class is called **arc** and it exposes all the functions that are capable of interacting with the remaining parts of Viz Arc as well as many helper functions. All of **arc**'s functions can be accessed via scripting by calling them directly, since they are all exposed directly to the global script, or via the **arc** keyword.

The following samples and codes snippets are all written using the **V8 JavaScript** syntax:

Accessing Viz Arc Functions

```
// Getting a reference to an action called VersusTemplate
```

```
var versus = arc.GetAction("VersusTemplate")
var versus = GetAction("VersusTemplate")
```

Note: You can also find this section in Viz Arc by selecting the **Help** Help button in the script section when in edit mode.

Action

All actions in the current project can be accessed using the **GetAction** method, whose content can be manipulated. See Action Properties for more details.

- BaseAction **GetAction** (string actionNameOrGUID)
  - Returns the first action found with the name provided. When a valid GUID is provided as a string, it returns the action found with the provided GUID.
- BaseAction **GetAction** (string actionName, string tabName)
  - Returns the first action found with the name provided inside the action tab named tabName.
- BaseAction **GetActionByName** (string actionName)
  - Returns the first action found with the name provided.
- BaseAction **GetActionByName** (string actionName, string tabName)
  - Returns the first action found with the name provided inside the action tab named tabName.
- BaseAction **GetActionByUUID** (Guid actionUUID)
  - Returns the first action found with the provided GUID.
- BaseAction[] **GetSelectedActions** ( )
  - Returns an array containing all the actions that are selected on the action canvas
- BaseAction[] **GetActionsOfTab** (string tabName, string actionType = "ALL")
  - Returns an array with all the actions inside the tab named *tabName* of type equal to the one provided in *actionType* input. Default value ("ALL") includes all actions found.
- BaseAction[] **GetActions** (string actionType = "ALL")
  - Returns an array with all the actions of the entire project of type equal to the one provided in *actionType* input. Default value ("ALL") includes all actions.
- BaseAction **GetChildAction** (string nameOrUUID)
  - Returns an action within the group, and only works on Group Actions.

GetAction Example

```
// Getting a reference to an action explicitly by its name "VersusTemplate"
var versus = GetActionByName("VersusTemplate")

// Getting a reference using a GUID
var versus = GetAction("e87a8031-a86b-4997-a169-c6f791920449")

// Getting a reference to an action called VersusTemplate
var versus = GetAction("VersusTemplate")

// Getting all NDI actions
var nidActions = GetActions("NDI")
```

Copyright © 2025 Vizrt

```
// get the action "PrecisionKeyer" within the "INITIALIZE" group action
var chromaAction = GetAction("INITIALIZE").GetChildAction("PrecisionKeyer")
```

Playlist

arc provides an alternative way of getting a BaseControlObject from a ControlObject/Blueprint action.

- void **ExecuteSelectedPlaylistRow** ( )
  - Executes the selected row on the playlist.
- void **ExecuteSelectedPlaylistRowAndNext** ( )
  - Executes the selected row on the playlist and changes selection to the next row.
- void **PreviewSelectedPlaylistRow** ( )
  - Previews the selected row on the playlist.
- BaseAction **GetSelectedPlaylistRowAction** ( )
  - Returns the action that's attached to the selected row on the playlist.
- void **SetSelectedPlaylistRow** (params string[] path)
  - Tries to find the row at path (path should contain a string per depth level) and makes it the selected row.
- void **PlayPlaylistByName** (string tabName)
  - Highlights the playlist with the name *tabName,* and plays it from the beginning.
- void **PlayPlaylistByIndex** (int tabIndex)
  - Highlights the playlist with 0-based index *tabIndex,* and plays it from the beginning.
- void **StopPlaylist** ( )
  - Stops the currently playing playlist.
- void **ChangePlaylistTab** (string tabName)
  - Selects and highlights the playlist with the name *tabName*.

Playlist Example

```
// Selects the row at "StatsDisplayGroup/AwayTeam/Show" and then previews and
executes it.
SetSelectedPlaylistRow("StatsDisplayGroup", "AwayTeam", "Show")
PreviewSelectedPlaylistRow()
ExecuteSelectedPlaylistRowAndNext()
```

Control Object

arc provides an alternative way of getting a BaseControlObject from a ControlObject/Blueprint action.

- BaseControlObject **GetControlObject** (ControlObjectAction action, string id)
  - Returns the control object with a specific ID from ControlObject action.

Getting a Specific ControlObject from a ControlObjectAction

```
// Get the ControlObject Action
```

Copyright © 2025 Vizrt

```
var MatchDayAction = GetAction("MatchdayTable")
// Get Title ControlObject (ControlText) and change its value
GetControlObject(Co, "Title").Value = "Sunday Fixtures"


// Get the Blueprint Action
var HeadlineBp = GetAction("HeadlineBp")
// Get Title ControlObject (String Variable) and change its value
GetControlObject(HeadlineBp , "Title").Value = "Lorem Ipsum"
```

MIDI

Attached and configured MIDI devices can be used to receive MIDI events using the **OnMIDIEvent** callback. It's also possible to send MIDI events to an attached device using the following methods:

- bool **SendMIDIControlMessage** (string DeviceName, int Channel, int Number, int Value)
  - Sends a MIDI control message to a the device named DeviceName, using Channel, Number and Value.
- bool **SendMIDINoteMessage** (string DeviceName, bool On, int Channel, int Note, int Velocity)
  - Sends a MIDI note message to a the device named DeviceName, using Channel, Note and Velocity. The parameter **On** determines whether the event is a note on or note off event.

 **Note:** Both of the methods above return true on successful completion and false if not successful.

MIDI Sample

```
Global.OnButtonPressed = function (id)
{
  SendMIDIControlMessage("Midi Fighter Twister", 1, 1, 127) // send control message
to Midi Fighter Twister on channel 1, number 1, value 127
  SendMIDINoteMessage("nanoPAD2", true, 1, 5, 100) // send note down event to
nanoPAD2 device
}

Global.OnMIDIEvent = function (midiEvent)
{
  // just print the midi event on the console
  Console.WriteLine("midi event |" + midiEvent.DeviceName + "| " +
midiEvent.EventType + "\n" + midiEvent.ToString())
}
```

Art-Net DMX

A sample on how to use the OnDMXEvent callback in a template or global script.

Art-Net Script Sample

```
Global.OnDMXEvent = function (dmxEvent)
```

Copyright © 2025 Vizrt

```
{
  Console.WriteLine( "Universe " + dmxEvent.Universe + " first change at channel " +
dmxEvent.firstDiff )
  Console.WriteLine( "Channel 7 has changed: " + dmxEvent.HasChanged(7) )
  Console.WriteLine( "Channel 7 value is: " + dmxEvent.DMXData[7])
}
```

You can enable or disable the DMX signals using the following methods

- void **EnableDMX** ( )
  Enables callbacks of the connected Art-Net devices to be sent to connected actions and script callbacks.
- void **DisableDMX** ( )
  Disables callbacks of the connected Art-Net devices to be sent to connected actions and script callbacks.
- bool **IsDMXEnabled** ( )
  Returns whether the Art-Net callbacks are enabled.

MQTT

Message Queuing Telemetry Transport is supported through the possibility to instantiate a MQTT client and send/receive messages

- ArcMqttClient **createMQTTClient** (string server, int port)Creates a MQTT client connected using server and port

The returned client ArcMqttClient supports the following methods

- void **Subscribe** (string topic, int qos = 1)
  Subscribes the client to the given topic with the specified quality of service (default 1).
- void **Unsubscribe** (string topic)
  Unsubscribes the client from the given topic.
- void **sendMessage** (string topic, string payload)
  Sends a message payloadto topic.
- void **Dispose** ( )
  Disconnects and deletes the client.

Whenever a message is received from a topic a client has subscribed to, the new data is set to the global DataMap using the topic as key and the payload as value. Payload data in JSON format is passed as a JSON object, anything else is passed as a string object.

MQTT Sample

```
var mqttClient

Global.OnInit = function ()
{
  mqttClient = createMQTTClient("localhost", 6548)
  mqttClient.Subscribe("hello/world/news")

  SubscribeDataMap("hello/world/news")
}
```

Copyright © 2025 Vizrt

```
Global.OnDataMapValueChanged = function (varName)
{
  if( varName == "hello/world/news" )
    Console.WriteLine("breaking news alert: " + GetData(varName))
}
```

A sample server written in C# illustrating the server side code using the MQTTnet library.

MQTT Server Sample

```
using MQTTnet;
using MQTTnet.Extensions.ManagedClient;
using MQTTnet.Server;
using System;
using System.Threading;

namespace testmqtt
{
    class Program
    {
        static void Main(string[] args)
        {
            var optionsBuilder = new MqttServerOptionsBuilder()
             .WithConnectionBacklog(100)
             .WithDefaultEndpointPort(6548);

            var mqttServer = new MqttFactory().CreateMqttServer();
            mqttServer.StartAsync(optionsBuilder.Build());

            int i = 0;
            MqttApplicationMessage message = null;
            while (true)
            {
                message = new MqttApplicationMessageBuilder()
                        .WithTopic("hello/world/news")
                        .WithPayload("Temperatures below " + i + " !")
                        .WithExactlyOnceQoS()
                        .Build();

                mqttServer.PublishAsync(message);
                Thread.Sleep(1000);
                Console.WriteLine(i + "");
                i--;
            }
        }
    }
}
```

Copyright © 2025 Vizrt

Object Tracker

The script exposes some useful functions that allows customization and remoting of the Object Tracker. For example, the **StopTracker** and **TakeOutTracker** function could be used to quickly remove tracking or On Air graphics.

- int **GetActiveTracker** ( )
    - Gets the currently active tracker index (starting from 1).
- int **SetActiveTracker** (int tracker)
    - Sets the currently active tracker index (starting from 1). Returns the active tracker index.
- void **TakeTracker** ( )
    - Takes tracker On Air all trackers.
- void **TakeOutTracker** ( )
    - Takes tracker Off Air all trackers.
- void **PreviewTracker** ( )
    - Previews all trackers.
- void **PreviewOutTracker** ( )
    - Removes all trackers from preview.
- void **StopTracker** ( )
    - Stops all trackers.
- void **StopTracker** (int index)
    - Stops the tracker with index (starting from 1).
- void **ResetPointerOffset**(int index)
    - Reset the pointer offset for tracker with index (starting from 1).

Viz Arena

The script exposes some useful functions concerning the **Viz Arena** integration.

- bool **DetectArenaCalibration** ( )
    - Redetects the camera calibration (same as the **D** shortcut in Viz Arena).
- bool **ClearArenaCalibration** ( )
    - Clears the camera calibration (same as the **BACKSPACE** shortcut in Viz Arena).
- bool **ClearArenaKeyer** ( )
    - Clears the Keyer mask (same as the **C** shortcut in Viz Arena).
- string[] **GetArenaCameraList** ( )
    - Returns a string-list of available cameras.
- string **GetCurrentArenaCamera** ( )
    - Returns the name of the current camera.
- int **GetCurrentArenaCameraIndex** ( )
    - Returns the zero based index of the current camera.
- bool **IsArenaConnected** ( )
    - Returns whether Viz Arena is running and connected to Viz Arc.

Parameter

All parameters are exposed to the global script and can be accessed via their unique ID.

Copyright © 2025 Vizrt

arc provides an alternative way of getting them.

- BaseParameter **GetParameter** (string id)
  - Gets the parameter identified by the unique id that was input.

It's also possible to get and set a parameter's value directly from **arc**.

- dynamic **GetParameterValue** (string id)
  - Gets the value of the parameter identified by the unique id that was input. [dynamic] The returned value's type depends on the parameter type.
- void **SetParameterValue** (string id, dynamic value)
  - Sets the value of the parameter identified by the unique id that was input. [dynamic] Input variable value can be of any type, see parameters for valid types.

In case you don't want the callback function **OnParameterChanged** to be triggered when changing a value using SetParameterValue, you can use **SetParameterRawValue**. This method does not trigger any calls to **OnParameterChanged**.

- void **SetParameterRawValue** (string id, dynamic value)
  - Sets the value of the parameter identified by the unique id that was input. [dynamic] Input variable value can be of any type, see parameters for valid types.

Buttons are a special case in the sense that they don't hold a value, and therefore have a separate method for triggering their *click*.

- void **PushParameterButton** (string id)
  - Triggers a pressed event on the button identified by the unique id that was input.

⚠ **Note:** Button presses trigger the global script's callback **OnButtonPressed.**

⚠ **Note:** Parameter value changes trigger the global script's callback **OnParameterChanged.**

Parameter Examples

```
// Setting the value of a bool parameter (id = ShowHighlights) to false
// direct assigment
ShowHighlights.Value = false
// Get parameter via arc and then assign to Value
GetParameter("ShowHighlights").Value = false
// Set Parameter value via arc without interacting with the actual parameter
SetParameterValue("ShowHighlights", false)

// Getting the value of a bool parameter (id = ShowHighlights)
var highlightState = GetParameterValue("ShowHighlights")

// Push LoadFixtures button
PushParameterButton("LoadFixtures")
```

Copyright © 2025 Vizrt

Channel

arc grants access to Viz Arc profiles. This is useful whenever more precise control is required for communicating with the Engines.

- ScriptingProfile **GetSelectedProfile** ( )
  - Returns the currently selected profile.
- int **GetChannelCount** ( )
  - Returns the number of channels on the currently selected profile.
- ScriptingChannel **GetChannel** (int index)
  - Returns the channel at the *index* position on the currently selected profile.
- ScriptingChannel **GetChannel** (string channelName)
  - Returns the channel named *channelName* on the currently selected profile.
- ScriptingChannel **GetPreviewChannel** ( )
  - Returns the preview channel of the currently selected profile.
- ScriptingChannel **GetProgramChannel** ( )
  - Returns the program channel of the currently selected profile.
- ScriptingChannel **GetSelectedChannel** ( )
  - In a template script it returns the currently selected channel of the Template Action.
    In the global script it returns the program channel of the currently selected profile.

Channel Handling Examples

```
// Clear main layer on all channels using GetChannelCount() and GetChannel(int)
for (var i = 0; i < GetChannelCount(); i++) {
  GetChannel(i).SendSingleCommand("RENDERER*MAIN_LAYER SET_OBJECT")
}

// Send message to VideoWallchannel via GetSelectedProfile () and GetChannel(string)
GetChannel("VideoWall").SendSingleCommand("RENDERER*MAIN_LAYER SET_OBJECT")
GetSelectedProfile().GetChannel("VideoWall").SendSingleCommand("RENDERER*MAIN_LAYER
SET_OBJECT")
```

Viz Engine/Unreal Engine Communication

arc provides quick access functions for sending messages to specific channels/Engines.

- void **SendSingleCommand** (string command, string channelName)
  - Sends *command* to all the Engines in the specified channel channelName.
- void **SendMultipleCommands** (string[] commands, string channelName)
  - Sends all the input *commands* to all the Engines in the specified channel *channelName*.
- string **GetFromEngine** (string command, string channelName)
  - Sends *command* to all the Engines in the specified channel *channelName*. Returns the answer to the sent *command*.
- string **GetFromVizEngine** (string command)
  - Sends command to the currently selected profile's Viz **editing Engine**. Returns the answer to the sent *command*.

Copyright © 2025 Vizrt

- string **GetFromUnrealEngine** (string command)
    - Sends command to the currently selected profile's Unreal **editing Engine**. Returns the answer to the sent *command.*
- string **GetFromEngineAsync** (string command, string channelName, int timeout = -1)
    - Sends *command* to all the Engines in the specified channel *channelName*. Returns the answer to the sent *command*.
      If timeout is specified and larger than 0, the method times out after *timeout* milliseconds if it does not receive an answer within that time.
- string **GetFromEngineAsync** (string command, int timeout = -1)
    - Sends *command* to all the Engines in the specified selected channel of the template. Returns the answer to the sent *command*.
      If timeout is specified and larger than 0, the method times out after *timeout* milliseconds if it does not receive an answer within that time.
- string **GetFromVizEngineAsync** (string command)
    - Sends command to the currently selected profile's Viz **editing Engine**. Returns the answer to the sent *command*.
- string **GetFromUnrealEngineAsync** (string command)
    - Sends command to the currently selected profile's Unreal **editing Engine**. Returns the answer to the sent *command*.

Viz Engine Communication

```
// Get scene from parameter and set it to Viz Engine main layer
SendSingleCommand(GetParameterValue("MainSceneSelector"), "Main")

// Clear Main, Back and Front layers on channel
var CleanCommands = ["RENDERER*MAIN_LAYER SET_OBJECT", "RENDERER*BACK_LAYER
SET_OBJECT", "RENDERER*FRONT_LAYER SET_OBJECT"]
SendMultipleCommands(CleanCommands, "Viz")

// Query Viz channel and Viz editing engine for the currently loaded scene
GetFromEngine("SCENE SCENE*SCENE GET", "Viz")
GetFromVizEngine("SCENE SCENE*SCENE GET")
```

The **async** variants can be used only when using the JavaScript language and have the advantage that they do not lock up the UI.

Async Samples

```
Global.OnButtonPressed = async function (id)
{
    if( id == "getVersionButton" ){
        const answer = await GetFromEngineAsync("VERSION", "localviz")
        Console.WriteLine("Viz Version is: " + answer)
    }
}
```

> ⚠ **Note:** If you use **await**, the enclosing function needs to be **async**. You can add this attribute manually in case you use it within a Viz Arc callback.

Tracking Hub Command

arc provides quick access functions for sending messages to the configured Tracking Hub.

- void **SendSingleTHCommand** (string command)
  - Sends *command* to the Tracking Hub (if configured and connected).
- string **GetFromTH** (string command)
  - Sends *command* to the Tracking Hub (if configured and connected) and returns the answer.
- string **GetFromTHAsync** (string command)
  - The asynchronous version of **GetFromTH**.

SMM Handling

- void **SendToSMM** (string key, string value, bool doEscape)
  - Sends key-value pair to Shared Memory to the first channel of the current profile. *doEscape* specifies whether the value string is escaped.
- void **SendToSMM** (string key, string value, bool doEscape, string channel)
  - Sends key-value pair to Shared Memory to all Engines contains in *channel*. *doEscape* specifies whether the value string is escaped.
- void **SendToSMM** (string key, string value, bool doEscape, string channel, string destination)
  - Sends key-value pair to Shared Memory to all Engines contains in *channel*. *doEscape* specifies whether the value string is escaped.
  - *destination* can be either SYSTEM, COMMUNICATION or DISTRIBUTED

The shared memory updates are sent to the UDP or TCP port configured on the target Viz Engine; if both are configured, it is sent to the UDP port. The Viz Communication Shared Memory map is therefore utilized. You can read more on Shared Memory configuration in the Profiles section in the *Viz Arc User Guide*.

SMM Example

```
// Send to Viz Channel SMM the variable "Target1" with the value from TargetState
SendToSMM("Target1", TargetState.Value, false, "Viz")

// Send to Viz Channel SMM the variable "Target2" with the value "Hello World!".
// The last parameter "DISTRIBUTED" indicates that the value will be propagated to
all engines connected to the same Graphic Hub
SendToSMM("Target2", "Hello World!", false, "Viz", "DISTRIBUTED")
```

GPI

The connected GPI state can be changed via the **arc** functionalities:

- void **SignalGpiChannel** (int channelIndex, bool signalHigh)
  - Signals set the GPI channel at *channelIndex* to either high or low.

Copyright © 2025 Vizrt

The following snippet presents a function that loads a scene to the "Main" channel and signals the GPI:

GetAction Example

```
function LoadScene()
{
    // Get scene from parameter and set it to Viz Engine main layer
    SendSingleCommand(GetParameterValue("MainSceneSelector"), "Main")
    // Set gpi channel 2 to High
    SignalGpiChannel(2, true)
}
```

Note: GPI must be enabled on the config.

Viz Pilot

Viz Pilot data elements can be created from scripting using the method *CreatePilotDataElement*. There are two ways to invoke this method:

- async bool **CreatePilotDataElement** (BaseAction action, string name)
    - Creates a Viz Pilot data element of *action* and name *name*. The function returns true on success.
- The BaseAction method:
    - async bool **CreatePilotDataElement** (string name)
        - Creates a Viz Pilot data element with name *name.* The function returns true on success.

```
Global.OnButtonPressed = async function (id)
{
    try{
      if( id == "createpilotButton" )
      {
          // use global method
          await CreatePilotDataElement(GetAction("loadSceneA"), "LoadSceneA")
          // use method defined on the BaseAction itself
          await
GetAction("loadSceneA").CreatePilotDataElement("LoadSceneA_fromAction")
      }
    }catch(ex)
    {
        // error handle
        Console.WriteLine("ex " + ex + " " + ex.stack)
    }
}
```

Timer

- void **CreateTimer** (string id)
    - Creates a timer that can be accessed via its unique *id*.

Copyright © 2025 Vizrt

- void **CreateTimer** (string id, int ms)
    - Starts a timer that can be accessed via its unique *id* and has a tick interval of *ms*.
- void **StartTimer** (string id, int ms)
    - Gets the timer identified by *id*, sets the tick interval to *ms* and starts it.
- void **StopTimer** (string id)
    - Gets the timer identified by *id* and stops it.

The following example creates a timer on the OnInit callback, makes use of two buttons to start/stop the timer and writes to the console whenever the timer ticks:

Timer Example

```
// Timer id
var heartBeatTimerId = "HeartBeat"

Global.OnInit = function ()
{
  // Create timer with id heartBeatTimerId
  CreateTimer(heartBeatTimerId)
}

Global.OnButtonPressed = function (id)
{
  if(id == "TimerStart")
    StartTimer(heartBeatTimerId, 1000)
  else if(id == "TimerStop")
    StopTimer(heartBeatTimerId)
}

// Script callback for timer ticks
Global.OnTimer = function (id)
{
  Console.WriteLine("Timer Tick " + id)
}
```

⚠ **Note:** Whenever a timer ticks the global script's callback **OnTimer** is called.

setTimeout

An alternative way to delay a certain action, is to use the built-in awaitable function *setTimeout*. It allows to call a function after a certain amount of time.

```
Global.OnButtonPressed = async function (id)
{
    if( id == "testButton" )
        await setTimeout(doSomething, 3000) // call doSomething after 3 seconds
}
```

Copyright © 2025 Vizrt

```
function doSomething()
{
    Console.WriteLine("we are in doSomething")
    SetData("timedate", Date().toString()) // set some data on the DataMap
}
```

In the example above, the method *doSomething* gets called after waiting for three seconds. It is important to use the **await** keyword before **setTimeout**. The await keyword can only be used in methods declared as **async**. Add the async keyword on the Viz Arc callbacks, as seen above with *Global.OnButtonPressed = async function (id)*.

StreamDeck

Any connected StreamDeck that is configured to be used exclusively with **arc**, can have its buttons customized using one of the following methods:

- void **SetStreamdeckKey** (int key, string label, int fontSize)
  - Baseline version, sets streamdeck key at *key index* image to a black square with *label* text of *fontSize* size.
- void **SetStreamdeckKey** (int key, string label, int fontSize, string imageFullPath)
  - Same as the baseline version but instead of a black block it sets a local image (at *imageFullPath*) as background. *imageFullPath* can be either a local file system path or a Graphic Hub path.
- void **SetStreamdeckKey** (int key, string label, int fontSize, int r, int g, int b)
  - Same as the baseline version but instead of black it uses an RGB color as background.
- void **SetStreamdeckKey** (int key, string label, int fontSize, int r, int g, int b, string imageName)
  - Same as baseline version using background color *r, g, b* and *imageName* on top of the background color (in case the image contains an alpha channel).
- void **SetStreamdeckKey** (int key, string label, int fontSize, string horAlignment, string vertAlignment, string textAlignment, int r, int g, int b, string imageName)
  - Same as the previous version, where text is *horAlignment* aligned horizontally, vertically by *vertAlignment* and the text itself is centerd through *textAlignment.*
    - hor*Alignment* can be either "Left", "Center" or "Right"
    - vert*Alignment* can be either "Top", "Center" or "Bottom"
    - *textAlignment* can be either "Left", "Center" or "Right"

Any key can have its contents cleared with the following method:

- void **ClearStreamdeckKey** (int key)
  - Clears the content of the Streamdeck key at key *index*

StreamDeck Key Configuration Example

```
function SetupStreamDeck()
{
    // Key 0: Black background, size 20 "Clear" text
    SetStreamdeckKey(0, "Clear", 20)
    // Key 1: Image background, size 20 "Load AR" text
    SetStreamdeckKey(1, "Load AR", 20, "D:/Soccer/Images/ARThumbnail.png")
    // Key 2: Blue background, size 20 "Continue" text
    SetStreamdeckKey(2, "Continue", 20, 0, 0, 255)
```

Copyright © 2025 Vizrt

```
    // Key 3: Gray background, using headshot from Graphic Hub (image may contain an
alpha channel)
    SetStreamdeckKey(3, " ", 20, 100, 100, 100, "IMAGE*/Default/MasterImages/
headshot_0123")
   // Key 4: Gray background, using headshot from Graphic Hub (image may contain an
alpha channel), Text "John Doe" is top left aligned
    SetStreamdeckKey(4, "John Doe", 20, "Left", "Top", "Left", 100, 100, 100,
"IMAGE*/Default/MasterImages/headshot_0123")
}

Global.OnInit = function () {
    // Clean first 3 keys
    ClearStreamdeckKey(0)
    ClearStreamdeckKey(1)
    ClearStreamdeckKey(2)

    SetupStreamDeck()
}
```

Another sample that prints some useful information using the **external StreamDeck plugin**.

```
function printSDEvenyInfo(sdEvent)
{
    // print all the available information for a StreamDeck event
    Console.WriteLine("StreamDeck Event Info:")
    Console.WriteLine("Event Type:   " + sdEvent.EventType)
    Console.WriteLine("Device Index: " + sdEvent.DeviceIndex)
    Console.WriteLine("Device ID:    " + sdEvent.Id )
    Console.WriteLine("Column:       " + sdEvent.XKey )
    Console.WriteLine("Row:          " + sdEvent.YKey )
    Console.WriteLine("Payload:      " + sdEvent.Payload )

    // if it's a dial or touch event print additional info
    // for StreamDeck + devices only
    if( sdEvent.hasOwnProperty("Ticks") )
        Console.WriteLine("Ticks:        " + sdEvent.Ticks )
    if( sdEvent.hasOwnProperty("TapPosX") )
        Console.WriteLine("TapPosX:      " + sdEvent.TapPosX )
    if( sdEvent.hasOwnProperty("TapPosY") )
        Console.WriteLine("TapPosY:      " + sdEvent.TapPosY )
}

Global.OnStreamDeckTouchTap = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)
}
Global.OnStreamDeckDialRotate = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)
}
Global.OnStreamDeckDialDown = function (sdEvent)
{
```

Copyright © 2025 Vizrt

```
    printSDEvenyInfo(sdEvent)
}
Global.OnStreamDeckDialUp = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)
}
Global.OnStreamDeckKeyDown = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)
}
Global.OnStreamDeckKeyUp = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)
}
```

Graphic Hub REST

arc provides some methods that allow to retrieve information about the current Graphic Hub REST server in use. It is meant to help using the Graphic Hub REST interface directly.

- string **GetGHHost** ( )
    - Returns the Graphic Hub REST host name (for example, *localhost or 10.81.44.71*).
- string **GetGHPort** ( )
    - Returns the Graphic Hub REST port (for example, `19398`).
- string **GetGHConnectionString** ( )
    - Returns the complete connection string based on the configured Host and Port (for example, *http://localhost:* `19398`).
- string **GetGHUser** ( )
    - Returns the Graphic Hub REST user name (for example, *Guest* or *Admin*).
- string **GetGHAuthenticationValue** ( )
    - Returns the base64 authentication string which is a combination of the user name and password (for example, *QWRtaW46Vml6RGI=*).
- bool **ImportArchive** (string path)
    - Import a via archive through the REST service. Beware that all assets in the via override the content's of the Graphic Hub. Returns true on success and false on failure.
- async Task<bool> **ImportArchiveAsync** (string path)
    - Async version of the above method.

Below is a code sample that fetches all the image names of a given Graphic Hub path using the **GetGHConnectionString** and **GetGHAuthenticationValue** functions.

Sample

```
function getFolderId( path )
{
    let folderId = ""
    let request = new XMLHttpRequest()
```

Copyright © 2025 Vizrt

```
    request.onreadystatechange = function() {
        if (request.readyState == 4 && request.status == 200 ) {
            //Console.WriteLine("respones: " + request.responseText)

            xmlDoc = new XmlDocument()
            xmlDoc.LoadXml(request.responseText)
            //Console.WriteLine("nodes " + xmlDoc.ChildNodes.Count)

            // create namespace manager
            nsmgr = new XmlNamespaceManager(xmlDoc.NameTable)
            // add namespace
            nsmgr.AddNamespace("x", "http://www.w3.org/2005/Atom")

            // search for x:model
            root = xmlDoc.DocumentElement
            folderId=root.SelectSingleNode("/x:feed/x:entry/x:id",
nsmgr).InnerXml.split(':')[2]
            Console.WriteLine("folder id " + folderId)
        }
    }
    request.open("GET", GetGHConnectionString()+"/translator/?path="+path, true)
    request.setRequestHeader("Authorization", "Basic " + GetGHAuthenticationValue())
    request.send();

    // fetch the images using the folder uuid
    GetImagesOfFolder(folderId)
}

function GetImagesOfFolder(folderId)
{
    let request = new XMLHttpRequest()

    request.onreadystatechange = function() {
        if (request.readyState == 4 && request.status == 200 ) {
            //Console.WriteLine("respones: " + request.responseText)

            xmlDoc = new XmlDocument()
            xmlDoc.LoadXml(request.responseText)
            //Console.WriteLine("nodes " + xmlDoc.ChildNodes.Count)

            // create namespace manager
            nsmgr = new XmlNamespaceManager(xmlDoc.NameTable)
            // add namespace
            nsmgr.AddNamespace("x", "http://www.w3.org/2005/Atom")

            // search for nodes 'entry'
            root = xmlDoc.DocumentElement
            imageNodes=root.SelectNodes("/x:feed/x:entry", nsmgr)
            //Console.WriteLine("nodes " + imageNodes.Count)

            let imageList = []

            for( node of imageNodes )
```

```
            {
                Console.WriteLine("image " + node.SelectSingleNode("./x:title",
nsmgr).InnerXml)
                imageList.push(node.SelectSingleNode("./x:title", nsmgr).InnerXml)
            }
            // set the dropdown
            // imagesDD.SetItems(imageList)
            }
    }
    request.open("GET", GetGHConnectionString()+"/files/" + folderId + "/?
term=IMAGE", true)
    request.setRequestHeader("Authorization", "Basic " + GetGHAuthenticationValue())
    request.send()
}
```

Other functions relative to the Graphic Hub:

- async Task<string[]> **GetImages**(path)
    - returns a string list of images in the Graphic Hub.

```
async function fetchImages(path)
{
    Console.WriteLine("fetching images in " + path)

    const results =  await GetImages(path).then(results =>
    {
        dropdown_0.SetItems(results)
    })
}

Global.OnButtonPressed = function (id)
{
    if( id == "button_0" )
    {
        dropdown_0.Clear()
        fetchImages("sports/soccer/headshots")
    }
}
```

The above code snippet populates a dropdown with the image names contained in the Graphic Hub path *sport/soccer/headshots*.

DataMap

arc provides a simple interface (get and set) for interacting with Viz Arc's DataMap:

- dynamic **GetData** (string varName)
    - Returns the value belonging to the variable named *varName.* [dynamic] Returned value depends on what was set to *varName*.
- void **SetData** (string varName, dynamic value)

Copyright © 2025 Vizrt

    - Inserts (or overwrites if varName already exists) t*he key:value* pair into Viz Arc's DataMap. [dynamic] Input *value* can be of any type.
- bool **HasData** (string varName)
    - Returns true if varName exists in the DataMap.
- void **SubscribeDataMap** (string variableName)
    - Subscribes to a specific key (Empty string subscribes to all changes). The subbed variables feedback triggers the script callback "OnDataMapValueChanged".
- void **UnsubscribeDataMap** (string variableName)
    - Unsubscribes from a specific key (Empty string unsubscribes to all changes).
- string[] **GetDataKeys** ( )
    - Returns a complete list of all DataMap key entries.

DataMap Example

```
Global.OnInit = function ()
{
  // make sure OnDataMapValueChanged is called when "someData" changes
  SubscribeDataMap("someData")

  // use blank string to subscrive to all DataMap changes
  //SubscribeDataMap("")

  // create a timer that triggers every second
  CreateTimer("aTimer")
  StartTimer("aTimer", 1000)
}

// Callback for DataMap changes
Global.OnDataMapValueChanged = function (varName)
{
  if(varName == "someData")
    UpdateSomeData(GetData(varName))
}

function UpdateSomeData( theData )
{
  // do something here
  Console.WriteLine( "new data " + theData )
}

Global.OnTimer = function (id)
{
  // generate some fresh data using the current time for testing
  // such that OnDataMapValueChanged gets called
  if( id == "aTimer" )
    SetData("someData", Date.now())
}

function printDataMap()
```

```
{
  var keys = GetDataKeys()

  for( k of keys )
    Console.WriteLine(k + " = " + GetData(k) )
}
```

Note: Whenever a DataMap variable changes the global script's callback **OnDataMapValueChanged** is called.

NDI

arc provides an interface to handle metadata feedback from NDI sources.

- string[] **GetNDISourceList** ( )
  - Returns an array with all the names of the available NDI sources.
- string[] **GetNDIPTZSourceList** ( )
  - Returns an array with all the names of the available NDI sources with PTZ control capabilities.
- void **SubscribeNdiSourceMetadata** (string source)
  - Subscribes to the metadata feedback on the NDI source identified by the provided source input. The feedback is sent to the datamap with key equal to the source name.
- void **UnsubscribeNdiSourceMetadata** (string source)
  - Unsubscribes the NDI feedback.
- bool **SendNDIMetadata** (string name, string XMLString)
  - Sends a **XMLString** to a source identified by **name**. Returns true on success, false otherwise.

DataMap Example

```
Global.OnInit = function ()
{
  // Get a list of available NDI sources (can take some time to update)
  var sources = GetNDISourceList()

  // subscribe to metadata changes on a ndi stream
  SubscribeNdiSourceMetadata("NEWTEKPTZ (Channel 1)")

  // metadata will be written into the DataMap, so register to the DataMap changes
also
  SubscribeDataMap("NEWTEKPTZ (Channel 1)")
}

Global.OnDataMapValueChanged = function (varName)
{
  Console.WriteLine(varName + " changed")
  // NDI metadata is typically in xml format
  Console.WriteLine(GetData(varName).ToString())
}
```

Copyright © 2025 Vizrt

```
Global.OnParameterChanged = function (id)
{
  if( id == "sendMetadata")
  {
    SendNDIMetadata("NEWTEKPTZ (Channel 1)", "<?xml version='1.0' encoding='UTF-8'?
><camera_control><command group_id='0' parameter_id='3' value='0.43'></
camera_control>")
  }
}
```

File Handling

- string **ReadTextFile** (string filename, string encoding = "UTF8")
  - Returns a *encoding* encoded string containing the whole content of the text file.
- bool **WriteTextFile** (string FullPath, string data, string encoding = "UTF8")
  - Writes a file at *FullPath* with its content equal to the encoded input *data*.

> ⚠ **Valid encodings:** "UTF8" , "ASCII", "BigEndianUnicode", "Default" [System defined encoding], "UTF32", "UTF7"

File Handling Example

```
// Get the StartList file content from the directory defined by the Directory
parameter "WorkingDir"
ReadTextFile( WorkingDir.Value + "\\StartList.json")

// Write the results to the directory defined by the Directory parameter "WorkingDir"
WriteTextFile( WorkingDir.Value + "\\RaceResults.json", results)
```

Logging

- bool **AddLog** (string fileName, string content)
  - Appends *content* to *fileName*, and returns true on success, false otherwise.
    If the file *fileName* does not exist, it is created.

```
// register data map changes somewhere (e.g. SubscribeDataMap(""))
Global.OnDataMapValueChanged = function (varName)
{
    AddLog("c:/tmp/templateLog.txt", "getting data " + varName + " = " +
GetData(varName))
}
```

The produced log file content contains entries as the example below:

```
2025/07/08 19:20:34.1389|getting data TIMECODE = 02:55:59:43
```

Copyright © 2025 Vizrt

JSON

- dynamic **ParseJson** (string data)
    - Deserializes the input *data* and returns a JSON object if successful.

On the returned JSON object you can access the members directly using their name. Use the "ToString()" method on any of the objects to convert them to strings.

```
var json = ParseJson("{time: '1994-11-05T13:15:30Z', title: 'Viz Arc', subtitle:
'Vizrt', messageId: 1}")
Console.WriteLine("the whole json " + json.ToString())
Console.WriteLine("the title is " + json.title.ToString())
```

When using the V8 Scripting Engine the built-in **JSON.parse** and **JSON.stringify** methods can be used

```
var json = JSON.parse('{"time": "1994-11-05T13:15:30Z", "title": "Viz Arc",
"subtitle": "Vizrt", "messageId": 1}')
Console.WriteLine("the whole json " + JSON.stringify(json))
Console.WriteLine("the title is " + json.title)
```

Excel

- string **convertXLSToCSV** (string excelFilePath, string csvOutputFile, string separator = "\t", int worksheetNumber = 1)
    - Converts an existing *.xls* file *excelFilePath* to a comma separated CSV file *csvOutputFile* usign *separator* (default tab separator) and using worksheet number *worksheetNumber* (1 default being the first worksheet in the Excel file).
    - On successful conversion the function returns the CSV output as a string.
- string **convertXLSToCSVString** (string excelFilePath, string separator = "\t", int worksheetNumber = 1)
    - Converts an existing *.xls* file *excelFilePath* to a comma separated CSV string using *separator* (default tab separator) and using worksheet number *worksheetNumber* (1 default being the first worksheet in the Excel file).
    - On successful conversion the function returns the CSV output as a stirng.
- void **convertXLSToCSVDataMap** (string excelFilePath, string dataMapPrefix, string separator = "\t", int fromSheet = 1, int toSheet = -1)
    - Converts an existing *.xls* file *excelFilePath* to a comma separated CSV file *csvOutputFile* usign *separator* (default tab separator) and using an optional range of worksheets. When *toSheet* is -1 it converts all worksheets. The resulting worksheets are written into the **DataMap** using the specified prefix in *dataMapPrefix.* The name of the worksheet is written to the DataMap key *dataMapPrefix<name>_<index>*.

```
let separator = ";"
// convert excel to CSV file, use ; as separator and read the second sheet
convertXLSToCSV("c:/tmp/ExcelData.xlsx", "c:/tmp/ExcelData.csv", separator, 2)
// read the whole csv file into a string
```

Copyright © 2025 Vizrt

```
var fileContent = ReadTextFile("c:/tmp/ExcelData.csv")
var EntryArr = fileContent.split("\n")

// First line is for the headers, ignore it
for(i = 0; i < EntryArr.length; i++)
{
    // split the row
    var spl = EntryArr[i].split(separator)
    if( spl.length <= 1 )
        continue;

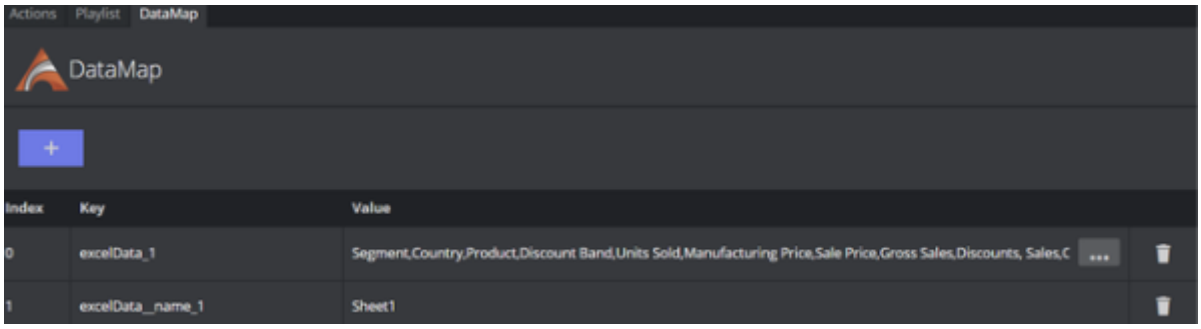
    Console.WriteLine("row " + i + ":")

    // print colums one by one separated by a whitespace
    for( entry of spl )
        Console.Write( entry.trim() + " " )

    Console.WriteLine("")
}

// another writing all worksheets to the DataMap, use separator ","
convertXLSToCSVDataMap("c:/tmp/ExcelData.xlsx", "excelData_", ",")
```

How the resulting **DataMap** might look like after calling *convertXLSToCSVDataMap* when the excel file contains just one worksheet.



Callbacks

- **OnParameterChanged** (string parameterID)
  - Called whenever a parameter (except button and table) changes. *parameterID* is the ID of the parameter that triggered the callback.
- **OnButtonPressed** (string buttonName)
  - Called when a parameter button is pressed. *buttonName* is the ID of the button that triggered the callback.
- **OnMiddleButtonPressed** (string buttonName)
  - Called when a button is pressed with the middle mouse button. *buttonName* is the ID of the button that triggered the callback.
- **OnRightButtonPressed** (string buttonName)
  - Called when a button is pressed with the middle mouse button. *buttonName* is the ID of the button that triggered the callback.
- **OnTimer** (string timerID)

Copyright © 2025 Vizrt

    - Called when a timer ticks (completes a cycle). *timerID* is the ID of the timer that triggered the callback.
- **OnDataMapValueChanged** (string varName)
    - Called whenever a DataMap variable changes. *varName* is the ID of the variable that was changed.
- **OnStreamDeckKey** (string key)
    - Called whenever a StreamDeck button is pressed. *key* indicates the index of the pressed button.
    - This callback is used in conjunction with the internal Stream Deck integration

The following StreamDeck callbacks are used with the external Stream Deck integration:

- **OnStreamDeckKeyUp** (sdEvent)
    - Called whenever a Stream Deck button has been released.
    - sdEvent contains the following fields:
        - string **EventType** (the event type, for example, "keyUp")
        - string **Id** (the unique ID of the device)
        - int **DeviceIndex** (the index assigned to this device)
        - int **XKey** (the column index of the key)
        - int **YKey** (the row index of the key)
        - string **Payload** (the user defined payload)
- **OnStreamDeckKeyDown** (sdEvent)
    - Called whenever a Stream Deck button has been pressed.
    - sdEvent contains the same fields as **OnStreamDeckKeyUp**
- **OnStreamDeckDialUp** (sdEvent)
    - Called whenever a Stream Deck button has been pressed.
    - sdEvent contains the same fields as **OnStreamDeckKeyUp**
- **OnStreamDeckDialDown** (sdEvent)
    - Called whenever a Stream Deck button has been pressed.
    - sdEvent contains the same fields as **OnStreamDeckKeyUp**
- **OnStreamDeckDialRotate** (sdEvent)
    - Called whenever a Stream Deck button has been pressed.
    - sdEvent contains the same fields as **OnStreamDeckKeyUp** and additionally:
        - int **Ticks** positive number for clockwise rotation and negative for anticlockwise rotation. Minimum value is 1 and increases when doing fast movementes.
- **OnStreamDeckTouchTap** (sdEvent)
    - Called whenever a Stream Deck button has been pressed.
    - sdEvent contains the same fields as **OnStreamDeckKeyUp** and additionally:
        - int **TapPosX** number from 0 to 200 (on Stream Deck + device) representing the horizontal touch position
        - int **TapPosY** number from 0 to 100 (on Stream Deck + device) representing the vertical touch position
- **OnMidiEvent** (midiEvent)
    - Called whenever a midi event is registered on one of the attached and configured midi devices.
    - midiEvent contains the following fields:
        - string **DeviceName** (the name of the device triggering the midi event.
        - string **EventType** (either "ControlChange", "NoteOn" or "NoteOff").
        - int **Channel** (the control channel of the event).
        - int **Number** (the control number of the event).

Copyright © 2025 Vizrt

        - int **Value** (the value of the event, in the range [0..127]).
        - int **Note** (the note of the event in case EventType is NoteOn or NoteOff).
        - int **Velocity** (the velocity of note event in case EventType is NoteOn or NoteOff).
- **OnDMXEvent** (dmxEvent)
    - Called whenever a dmx lighting value changes
    - dmxEvent contians the followinf fields:
        - short **Universe** (the Univers ehtat changed)
        - byte[] **DMXData** (the entire 512 byte long data array)
        - byte[] **change** (a 512 byte long array containing information about channel changes)
        - int **firstDiff** (the index of the first channel that changed)
    - bool HasChanged(int index)
      Call this function to check whether a certain channel has changed
- **Table Callbacks**
    - **OnTableColumnsChanged** (string tableID)
        - Called whenever a table parameter's columns change in number. *tableID* is the ID of the table that triggered the callback.
    - **OnTableRowsChanged** (string tableID)
        - Called whenever a table parameter's rows change in number. *tableID* is the ID of the table that triggered the callback.
    - **OnTableCellValueChanged** (string tableID, int row, int column, BaseBlock cell)
        - Called whenever a table parameter's cell changes value. *tableID* is the ID of the table that triggered the callback. *row* and *column* indicate the position of the cell within the caller table parameter. *cell* is the cell object that was changed. Users can interact directly with it. When **Trigger on live changes** is enabled on the table property, this callback is called also while editing the cell, if not the callback is called when keyboard focus is lost on the edited cell.

Video Output Callbacks

- **OnVideoMouseLeftButtonDown** (Point point)
    - Invoked when the left mouse button is pressed on the preview output. *point.X* and *point.Y* are normalized coordinates in the range of [-0.5…0.5] where [0,0] is the centre of the screen.
- **OnVideoMouseRightButtonDown** (Point point)
    - Invoked when the right mouse button is pressed on the preview output. *point.X* and *point.Y* are normalized coordinates in the range of [-0.5…0.5] where [0,0] is the centre of the screen.
- **OnVideoMouseMove** (Point point)
    - Invoked when the right mouse moves on the preview output. *point.X* and *point.Y* are normalized coordinates in the range of [-0.5…0.5] where [0,0] is the centre of the screen.
- **OnVideoMouseWheel** (double delta)
    - Invoked when the mouse wheel is turned on the preview output. delta is a floating point typically being -120/120 depending on the direction of the wheel turn and the hardware connected.
- **OnVideoKeyDown** (char charKey , uint rawKey , KeyEventArgs eventArgs)
    - Invoked when a keyboard down event has taken place on the preview output. *charKey* contains the actual character of the pressed key, *rawKey* is the numeric representation of the pressed key and *eventArgs is a* System.Windows.Input.KeyEventArgs instance from the operating system representing the raw event information.

Copyright © 2025 Vizrt

Exposed Objects

Console

- void **Write** (string message)
  - Writes the message to the scripting console.
- void **WriteLine** (string message)
  - Writes the message to the scripting console followed by a new line.

> **Info:** When Viz Arc's log level is set to **TRACE**, the strings sent to *Write* and *WriteLine* are also logged in the global log file.

MessageBox

- void **Show** (string message)
  - Shows a message box with its content equal to *message*.
- void **Show** (string message, string title)
  - Shows a message box with titled *title* and with its content equal to *message*.

File Handling Example

```
// Log an error and show a message to the user
Console.WriteLine("Unable to load data")
MessageBox.Show("Unable to load data", "Load Error")
```

XmlDocument

XmlDocument allows you to read XML files or strings and aggregate data using XPath. Read more about XMLDocument and other classes here.

```
// create XmlDocument and load a xml from disc
xmlDoc = new XmlDocument()
xmlDoc.Load("C:/tmp/TestData.xml")
Console.WriteLine("nodes " + xmlDoc.ChildNodes.Count)


// create namespace manager
nsmgr = new XmlNamespaceManager(xmlDoc.NameTable)
// add namespace
nsmgr.AddNamespace("x", "http://www.contoso.com/books")

// search for book nodes under the books node
root = xmlDoc.DocumentElement
nodeList=root.SelectNodes("/x:books/x:book", nsmgr)
Console.WriteLine("books " + nodeList.Count)

for( var book of nodeList )
```

Copyright © 2025 Vizrt

```
    Console.WriteLine("ISBN: " + book.GetAttribute("ISBN") + " title: " +
book.SelectSingleNode("./x:title", nsmgr).InnerXml)
```

The content of the sample test file *C:/tmp/TestData.xml* might look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<books xmlns="http://www.contoso.com/books">
  <book genre="novel" ISBN="1-861001-57-8" publicationdate="1823-01-28">
    <title>Pride And Prejudice</title>
    <price>24.95</price>
  </book>
  <book genre="novel" ISBN="1-861002-30-1" publicationdate="1985-01-01">
    <title>The Handmaid's Tale</title>
    <price>29.95</price>
  </book>
  <book genre="novel" ISBN="1-861001-45-3" publicationdate="1811-01-01">
    <title>Sense and Sensibility</title>
    <price>19.95</price>
  </book>
</books>
```

XMLHttpRequest

With the XMLHttpRequest class you can fetch data from a remote server. Below is a sample that fetches asynchronously JSON data from a server.

```
var request = new XMLHttpRequest()

request.onreadystatechange = function() {
  if (request.readyState == 4 && request.status == 200 ) {
    Console.WriteLine("we are here")

    var json = JSON.parse(request.responseText)

    Console.WriteLine(JSON.stringify(json))

    for (elem of json)
        Console.WriteLine(elem.name)
  }
}
request.open("GET", "https://jsonplaceholder.typicode.com/users", true)
request.setRequestHeader( "Content-Type", "application/json" ) // make sure the
request header is set AFTER calling open
request.send()
```

Prevent Caching

It is possible that requests through **XMLHttpRequest** get cached and the results of the queries might seem outdated. In order to prevent caching you can add a random number as parameter of the request.

Copyright © 2025 Vizrt

`request.open("GET", "https://jsonplaceholder.typicode.com/users?dummy="+Date.now() , true)`
In this case a dummy parameter is assigned with the current EPOCH date in milliseconds.

xlAppType

This type allows you to read Excel sheets directly.

> **Note:** This object only works if there is a local Excel installation on the same machine where Viz Arc is running.

FSO

The FSO object allows you to read, create and write files.

- **OpenTextFile** (*filename*, [*iomode*, [*create*, [*format*]]])
  - *iomode* can be one of the following: IOMode.ForReading, IOMode.ForWriting, IOMode.ForAppending
  - *format* can be of the following: Tristate.TristateUseDefault (system default), TriState.TristateTrue (Unicode), TriState.TristateFalse (ASCII).

Reading a UTF8 Encoded Text File

```
var file = new FSO()
var stream = file.OpenTextFile("d:/testexport.txt")
// or
var stream = file.OpenTextFile("d:/testexport.txt", IOMode.ForReading, false,
Tristate.TristateTrue)

Console.WriteLine(stream.ReadAll())
```

xHost

The xHost object gives you access to virtually any .NET resource.

V8 Script Sample

```
var List = xHost.type('System.Collections.Generic.List')
var DayOfWeek = xHost.type('System.DayOfWeek')
var week = xHost.newObj(List(DayOfWeek), 7)
week.Add(DayOfWeek.Sunday)
```

You can even import entire assemblies:

Copyright © 2025 Vizrt

V8 Enumerate Files in Directory

```
var clr = xHost.lib('mscorlib', 'System', 'System.Core', 'System.IO')
dropdown_0.Clear()
var dir = clr.System.IO.Directory
dropdown_0.SetItems(dir.GetFiles('c:\\tmp'))

// another sample that starts an external process "calc.exe"
var proc = xHost.lib('System.Diagnostics.Process')
proc.System.Diagnostics.Process.Start("calc.exe")
```

In the example above, a UI dropdown element named *dropdown_0* is populated with a file list contained in *c:\tmp* using .NET **System.IO.Directory** class instance.

Performance

The performance object provides access to performance-related information.

- **now** ( )
  Returns a floating point value of the EPOCH time in milliseconds.
- **sleep** (milliseconds, precise)
  Sleep for a certain amount of milliseconds. Set precise to **true** when a high precision timer is required.

```
let timeA = Performance.now()
Performance.sleep(500, true) // high perf sleep half second
// do something else
let timeB = Performance.now()

Console.WriteLine("timeA " + timeA)
Console.WriteLine("timeB " + timeB)
Console.WriteLine("time difference " + (timeB- timeA) + " ms")
```

SQL Sample

If you know the assembly name of a specific type, you can instantiate it using xHost.type(name, assemblyName) method.

```
var queryString = "SELECT * FROM someTable"
var connetionString = "Server=192.168.1.42,1433;UID=aUserID;PWD=SomePassword;"

// get types using xHost.type
var SqlConnection = xHost.type('System.Data.SqlClient.SqlConnection',
'System.Data.SqlClient')
var SqlCommand = xHost.type('System.Data.SqlClient.SqlCommand',
'System.Data.SqlClient')
var SqlDataReader = xHost.type('System.Data.SqlClient.SqlDataReader',
'System.Data.SqlClient')
```

Copyright © 2025 Vizrt

```
connection = new SqlConnection(connetionString)
connection.Open()
var command = new SqlCommand(queryString, connection)
var reader = command.ExecuteReader()
// do something with the data
while (reader.Read())
{
    // iterate over the result and print the result on the console
    Console.WriteLine(reader.GetString(0))
}
connection.Close()
```

SQLite Sample

Another sample that uses the xHost.type function is the usage of a simple SQLite database. It is required that the SQLite libraries/DLLs are in the search path of the system or in the same directory as the Viz Arc executable.

```
var SQLiteConnection = xHost.type('System.Data.SQLite.SQLiteConnection',
'System.Data.SQLite')
// those two below are not needed for this sample as they dont get instantiated
explicityl
var SQLiteDataReader = xHost.type('System.Data.SQLite.SQLiteDataReader',
'System.Data.SQLite')
var SQLiteCommand = xHost.type('System.Data.SQLite.SQLiteCommand',
'System.Data.SQLite')

function ReadData(conn)
{
    // create q query
    sqlite_cmd = conn.CreateCommand()
    sqlite_cmd.CommandText = "SELECT Name FROM Artist LIMIT 10;" // read first 10
artists of the table

    // execute the query
    sqlite_datareader = sqlite_cmd.ExecuteReader();
    while (sqlite_datareader.Read())
    {
        // iterate over the result and print the result on the console
        Console.WriteLine(sqlite_datareader.GetString(0));
    }
}

function testSQLiteDB()
{
    // create a new connection specifying the database file name and the version
    // sample database can be found here https://github.com/lerocha/chinook-database
    conn = new SQLiteConnection("Data Source=C:\\tmp\\Chinook.db;Version=3;")
    try
    {
        // Open the connection:
```

```
        conn.Open()

        // read some data
        ReadData(conn)

        // close the connection
        conn.Close()
    }
    catch (ex)
    {
        Console.WriteLine("error in SQLite query " + ex)
    }
}

Global.OnInit = function ()
{
    testSQLiteDB()
}
```

TcpSend

The TcpSendAsync method, is used to send a short message to any host on any port of the network.

- Task<string> **TcpSendAsync** (string hostname, int port, string command, int timeoutInMs = 1000)
    - Send *command* to *hostname* on *port*. If the parameter *timeoutInMs* is not specified, a default timeout of 1 second is used.

```
Global.OnButtonPressed = async function (id)
{
    if( id == "sendTcp"){
        let response await TcpSendAsync("192.168.1.22", 1234, "say something\0",
5000)
        Console.WriteLine("response from client " + response)
    }
}
```

HtmlAgility Example

The classes **HtmlDocument** and **HtmlWeb** are exposed by the HtmlAgility library and enable parsing and data extraction of html pages.

```
//sample to use HtmlDocument class from HtmlAgility
 var doc = new HtmlDocument()
 doc.Load("c:/tmp/HtmlAgilityTest.html")

 for (var table of doc.DocumentNode.SelectNodes("//table")) {
   Console.WriteLine("Found: " + table.Id)

   for (var row of table.SelectNodes("tr")) {
```

Copyright © 2025 Vizrt

```
      for (var cell of row.SelectNodes("th|td"))
        Console.Write(cell.InnerText + " ")
      Console.WriteLine("")
    }
  }

  // sample to use HtmlWeb class from HtmlAgility
  var html = "http://html-agility-pack.net/"
  var web = new HtmlWeb()
  var htmlDoc = web.Load(html)
  var node = htmlDoc.DocumentNode.SelectSingleNode("//head/title")
  Console.WriteLine("Node Name: " + node.Name + "\n" + node.OuterHtml)
```

The contents of the file from the sample above *c:/tmp/HtmlAgilityTest.html*:

HTML Sample with Table

```
<!DOCTYPE html>
<html>
<style>
    table, th, td {
        border: 1px solid black;
    }
</style>
<body>
    <h2>A HTML table to test HTML Agility</h2>
    <table style="width:100%" id="dataTable">
        <tr>
            <th>Name</th>
            <th>Number</th>
            <th>Country</th>
        </tr>
        <tr>
            <td>Athlete A</td>
            <td>42</td>
            <td>Italy</td>
        </tr>
        <tr>
            <td>Athlete B</td>
            <td>34</td>
            <td>Japan</td>
        </tr>
    </table>
    <p>Here is some more text</p>
</body>
</html>
```

Read more about HtmlAgility here.

Copyright © 2025 Vizrt

Main Script-only

There are functionalities that are specific to Viz Arc's main script:

Canvas Tabs Handling

- void **SetActionsSelectedTab** (string tabName)
  - Looks for a tab named *tabName* and sets it as active.
- void **SetActionsSelectedTab** (int tabIndex)
  - Sets the Action selected tab to the tab at *tabIndex* index.
- string **GetActionsSelectedTabName** ( )
  - Returns the currently selected tab's name.
- string[] **GetActionsTabs** ( )
  - Returns a string array with all tab names.

Action Template Handling

Arc's main scripts allows the user to interact with template actions on the action canvas.

- void **PreviewSelectedTemplate** ( )
  - Previews the currently selected template action.
- void **ExecuteSelectedTemplate** ( )
  - Executes the currently selected template action.
- void **UpdateSelectedTemplate** ( )
  - Updates the currently selected template action.
- void **ContinueSelectedTemplate** ( )
  - Continues the currently selected template action.
- void **StillPreviewSelectedTemplate** ( )
  - Generates a still preview of the currently selected template action.

> ⚠ **Note:** The method presented only works when one and only one action (template action) is selected on the action canvas.

Callbacks

- **PreActionExecute** (string actionName)
  - Called whenever an is executed and before the actual execution occurs.
  - actionName is the name of the action that is being executed.
- **PosActionExecute** (string actionName)
  - Called whenever an is executed and after the actual execution occurs.
  - actionName is the name of the action that is being executed.
- **OnInit** ( )
  - Called when the main script is started (User clicks on the **Start** button).

Template Script-only

The template script is a specific version that is used on the template designer and on the template action.

Copyright © 2025 Vizrt

Action/Designer Handling

- void **ExecuteTemplate** ( )
  - Execute the owner template action or loaded template in the designer.
- void **ContinueTemplate** ( )
  - Continues the owner template action or loaded template in the designer.
- void **OutTemplate** ( )
  - Takes out the owner template action or loaded template in the designer.
- void **UpdateTemplate** ( )
  - Updates the owner template action or loaded template in the designer.
- void **UpdateTemplate** (string COs = null)
  - Updates the owner template action or loaded template in the designer. The parameter COs is a space separated list of Control Object ID's that shall be updated. For large templates containing a big amount of ControlObjects this is a very efficient alternative whenever only a small part in the scene needs to be updated. For example, UpdateTemplate("currentScore totalScore") updates only the two ControlObjects with id "currentScore" and "totalScore".
- void **UpdateTemplate** ("BP_Name.VariableName")
  - Updates the owner template action or loaded template in the designer. This method can be used if the scene is in Unreal Engine. For large templates containing a large amount of variables, this is a very efficient alternative when only a variable in the blueprint needs to be updated.
- void **UpdateTemplate** ("BP_Name", "[VariableName1, VariableName2, ecc...]")
  - Updates the owner template action or loaded template in the designer. This method can be used if the scene is in Unreal Engine. For large templates containing a large amount of variables, this is a very efficient alternative when only a few variables in the blueprint need to be updated.
- void **PreviewTemplate** ( )
  - Previews the owner template action or loaded template in the designer.
- void **PreviewExecuteTemplate** ( )
  - Executes the owner template action or loaded template in the designer to the preview channel.
- void **PreviewContinueTemplate** ( )
  - Executes the owner template action or loaded template in the designer to the preview channel.
- void **PreviewOutTemplate** ( )
  - Executes the owner template action or loaded template in the designer to the preview channel.
- void **PreviewUpdateTemplate** (string COs = null)
  - Updates the owner template action or loaded template in the designer to the preview channel. The parameter COs is a space separated list of Control Object ID's that shall be updated. For large templates containing a big amount of ControlObjects this is a very efficient alternative whenever only a small part in the scene needs to be updated. For example, UpdateTemplate("currentScore totalScore") updates only the two ControlObjects with id "currentScore" and "totalScore".

Properties

- BaseAction **ThisAction**
  - Returns the script accessor of this template. **ThisAction** might be null when using the template editor.

Copyright © 2025 Vizrt

```
if( ThisAction ){
  // set the action's name
  ThisAction.Name = "Hello"
  // set the action's tooltip description
  ThisAction.Description = "Hello Description"
}
```

Control Object Handling

The template script allows the user to interact with the template's control objects.

- void **SetControlObject** (string objectID, dynamic value)
  - Sets the control object with id equal to *objectID's* value to *value*. The set object's value is sent on template execute/update.
  - ControlObject ID's set by this methods which have not been present in the payload during template creation, are added dynamically.

⚠ **Note:** SetControlObject only works on control objects that aren't already linked to parameters.

Template Channels Handling

The template script allows the user to change the program output channel.

- void **SetSelectedChannel** (string name)
  - Sets the selected program output channel to *name*.
- ScriptingChannel **GetSelectedChannel** ( )
  - Returns the currently selected program channel of the template action.

ScriptingChannel

The ScriptingChannel class is used for finer control on the Engines contained in the channel.

Properties

- **Name**
  - The channel's name.
- **Count**
  - The number of Engines in the channel.

Methods

- void **SendSingleCommand** (string command)
  - Sends *command* to all the Engines in the channel.
- void **SendMultipleCommands** (string[] commands)
  - Sends all the input *commands* to all the Engines in the channel.
- void **SendToSMM** (string key, string value, bool doEscape)
  - Sends key-value pair to Shared Memory to all Engines contained in the channel. doEscape specifies whether the value string is escaped.

Copyright © 2025 Vizrt

- void **SendToSMM** (string key, string value, bool doEscape, string destination)
    - Sends key-value pair to Shared Memory to all Engines contained in the channel. doEscape specifies whether the value string is escaped.
    - *destination* can be either SYSTEM, COMMUNICATION or DISTRIBUTED

Template Scene Handling

The template script allows the user to set the scene that should be loaded when executing.

- void **SetSceneFullpath** (string fullpath = null)
    - The input fullpath is the value that is sent to the Engine when executing the template. When no full path is provided the user config value is removed and the original attached scene is used.
- string **GetBaseContainerPath** ( )
    - Returns the current base container path where the root control object is located.
- bool **SetBaseContainerPath** ( )
    - Sets the base container path where the root control object is located. This might be useful to redirect the destination of the ContolObjects to a different container in the same scene.
- string **GetDirector** ( )
    - Gets the current director executed on **Execute** or on **Continue**.
- bool **SetDirector** (string dir)
    - Sets the director executed on **Execute** or on **Continue**.

Template Action Configuration

- bool **IsCommandHeaderVisible**
    - Indicates whether the CommandHeader should be visible on the template action.
- bool **UpdateOnSelected**
    - When this flag is set, the script callbacks are only triggered when the template action is selected on the action canvas (blue border).

Callbacks

- **OnCreated** ( )
    - Called when the template script is executed (when the template action is created and when the template opened on the designer is started).
- **OnShow** ( )
    - Called when the template is shown (when the template action's pop-up is opened, when the action becomes embedded and when the template opened on the designer is started).
- **OnExecute** ( )
    - Called when the template is executed.
- **OnPreviewExecute** ( )
    - Called when the template is executed to the preview channel.
- **OnContinue** ( )
    - Called when the template is continued.
- **OnPreviewContinue** ( )
    - Called when the template is continued to the preview channel.
- **OnUpdate** ( )

Copyright © 2025 Vizrt

    - Called when the template is updated.
- **OnPreviewUpdate** ( )
    - Called when the template is updated to the preview channel.
- **OnOut** ( )
    - Called when the template is taken out.
- **OnPreview** ( )
    - Called when the template is previewed.
- **OnTrackerAction** (string action)
    - Called on certain Object Tracker events. the *action* parameter determines the type of event:
        - **take:** Triggered when Object Tracker is taken On Air.
        - **takeout:** Triggered when Object Tracker is taken Off Air.
        - **preview:** Triggered when Object Tracker preview is taken.
        - **previewout:** Triggered when Object Tracker preview is taken out.
        - **newTracker <index>:** Triggered whenever a new object has been selected for tracking. *index* is 1 based.
        - **lostTracker <index>:** Triggered whenever a tracked object has lost tracking. *index* is 1 based.

Sample Usage of Object Tracker Script API

```
Global.OnTrackerAction = function (action)
{
    Console.WriteLine("tracker action " + action )

    if( action == "take" )
        GetAction("DATA").Execute()
    else if( action.startsWith("newTracker" ) ){
        // we want to take off air whatever is On Air when we select a new tracked
object
        TakeOutTracker()
        Console.WriteLine("OFF AIR" )
    }
}
```

- **OnArenaPosition** (double screenX, double screenY, double worldX, double worldY, double worldZ)
    - Called when the user clicks on the Viz Arena view with the positioning tool.
        - **screenX** and **screenY** are the screen coordinates of the mouse click. The lower left corner of the arena screen is the origin (0,0).
        - **wolrdX**, **worldY** and **worldZ** are Viz Engine world coordinates, the units (default meters) are the same as for the selected **Viz Arena** project.

Common Callbacks

Callbacks that can be used in the global script and template scripts

- **OnVideoMouseLeftButtonDown** (point)
    - Called when the user pressed the left mouse button on the video output.
        - **point.X** normalized value in the range [-0.5, 0.5]
        - **point.Y** normalized value in the range [-0.5, 0.5]

Copyright © 2025 Vizrt

- **OnVideoMouseRightButtonDown** (point)
    - Called when the user pressed the left mouse button on the video output.
        - **point.X** normalized value in the range [-0.5, 0.5]
        - **point.Y** normalized value in the range [-0.5, 0.5]
- **OnVideoMouseMove** (point)
    - Called when the user moved the mouse on the video output.
        - **point.X** normalized value in the range [-0.5, 0.5]
        - **point.Y** normalized value in the range [-0.5, 0.5]
- **OnVideoMouseWheel** (delta)
    - Called when the user rolled the mouse wheel on the video output.
        - **delta** is a floating point value, typical values are -120.0 for mouse wheel down and 120.0 for mouse wheel up.

Parameters

Parameters are the base components of Viz Arc's scripting. A list of all existing parameters types and their associated properties is presented below.

Base Parameters Functionality

The following properties and methods are shared among all parameters

- string **Label** [Get, Set]
    - Gets/sets the label that is displayed on the UI.
- bool **IsEnabled** [Get, Set]
    - Gets/sets the enabled status of the parameter. Disabled parameters can be interacted with.
- bool **IsVisible** [Get, Set]
    - Whether the parameter is visible. Invisible parameters are visible (displayed as grayed out) only while editing (and scripts are not running).
- double **X** [Get, Set]
    - Gets/sets the horizontal position of the parameter on the canvas.
- double **Y** [Get, Set]
    - Gets/sets the vertical position of the parameter on the canvas.
- double **Width** [Get, Set]
    - Gets/sets the width of the parameter.
- double **Height** [Get, Set]
    - Gets/sets the height of the parameter.
- void **SetColor** (byte r, byte g, byte b, byte a = 255)
    - Sets the parameter's color to the input RGBA color.
- string **Color**
    - Gets/sets the parameter's selected color in Hex format, for example, #FF0A0A8C (#RRGGBBAA).
- int **ColorR**
    - Gets/sets the parameter's selected red color value in the range **[0, 255]**.
- int **ColorG**
    - Gets/sets the parameter's selected green color value in the range **[0, 255]**.
- int **ColorB**
    - Gets/sets the parameter's selected blue color value in the range **[0, 255]**.

Copyright © 2025 Vizrt

- int **ColorA**
    - Gets/sets the parameter's selected alpha value in the range **[0, 255]**.
- string **Tooltip** [Get, Set]
    - Gets/sets the tooltip of the UI element.
- string **LinkedCO** [Get]
    - The name of the associated ControlObject (Template Scripting only).
- void **UpdateDataLink**()
    - Forces an explicit evaluation of the DataLink expression associated with this parameter.

A sample use of the **LinkedCO** property:

```
Global.OnParameterChanged = function (id)
{
    // get linked ControlObject id associated to parameter 'id'. It is null if it's
not linked to any CO.
    let linkedCO = GetParameter(id).LinkedCO

    if( linkedCO )
    {
        Console.WriteLine("Changed: " + id + ", Linked Control Object ID: " +
linkedCO)

        // live update the template
        UpdateTemplate(linkedCO)
    }
}
```

Layout

The layout parameters allow the user to organize and improve the usability of a script/template.

Panel

- BaseParameter [ ] **Children** [Get]
    - Returns an array with all of the panel's children.
- BaseParameter **GetParameter** (string parameterID)
    - Tries to find a child with id equal to *parameterID*. Returns it if successful.

Tabs

- string **Value** [Get, Set]
    - Set: Attempts to find a tab with its name equal to the input. If found, sets it as selected tab.
    - Get: Returns the name of the selected tab.
- BaseParameter [ ] **Children** [Get]
    - Returns an array with all of the panel's children.
- BaseParameter **GetParameter** (string parameterID)
    - Tries to find a child with id equal to *parameterID*. Returns it if successful.
- bool **AllowReordering**
    - Whether a user can reorder the tabs.

Copyright © 2025 Vizrt

- int **SelectedIndex** [Get, Set]
  - Gets/sets the index of the selected tab.

Info

- string **Value** [Get, Set]
  - Gets/sets the info text that displays on the parameter.

Label

TextColor

- string **Value** [Get, Set]
  - Gets/sets the labels text color in Hex format, for example, #FF0A0A8C (#RRGGBBAA).

Dialogs

Color

- string **Value** [Get, Set]
  - Gets/sets the parameter's selected color in Hex format, for example, #FF0A0A8C (#RRGGBBAA).
- int **R** [Get]
  Gets the value of the red component in the range **[0, 255]**.
- int **G** [Get]
  Gets the value of the green component in the range **[0, 255]**.
- int **B** [Get]
  Gets the value of the blue component in the range **[0, 255]**.
- int **A** [Get]
  Gets the value of the alpha component in the range **[0, 255]**.
- int **RPercent** [Get]
  Gets the value of the red component in the range **[0, 1]**.
- int **GPercent** [Get]
  Gets the value of the green component in the range **[0, 1]**.
- int **BPercent** [Get]
  Gets the value of the blue component in the range **[0, 1]**.
- int **APercent** [Get]
  Gets the value of the alpha component in the range **[0, 1]**.
- void **SetR** (int R)
  Sets the red component in the range **[0, 255]**.
- void **SetG** (int G)
  Sets the green component in the range **[0, 255]**.
- void **SetB** (int B)
  Sets the blue component in the range **[0, 255]**.
- void **SetA** (int A)
  Sets the alpha component in the range **[0, 255]**.
- void **SetRGB** (int R, int G, int B)
  Sets the red, green and blue components in the range **[0, 255]**.

Copyright © 2025 Vizrt

- void **SetRGBA** (int R, int G, int B, int A)
  Sets the red, green, blue and alpha components in the range **[0, 255]**.
- void **SetRPercent** (double R)
  Sets the red component in the range **[0, 1]**.
- void **SetGPercent** (double G)
  Sets the green component in the range **[0, 1]**.
- void **SetBPercent** (double B)
  Sets the blue component in the range **[0, 1]**.
- void **SetAPercent** (double A)
  Sets the alpha component in the range **[0, 1]**.
- void **SetRGBPercent** (double R, double G, double B)
  Sets the red, green and blue components in the range **[0, 1]**.
- void **SetRGBAPercent** (double R, double G, double B, double A)
  Sets the red, green, blue and alpha components in the range **[0, 1]**.

Directory

- string **Value** [Get, Set]
  - Gets/sets the selected directories fullpath.
  - Set: The input value must be a valid directory in the file system.

File

- string **Value** [Get, Set]
  - Gets/sets the selected file's fullpath.
- bool **WatchFile** [Get, Set]
  - Enable/disable the watchfile feature.
- bool **ReadRawContent** [Get, Set]
  - Set to true if the raw file content should be read into the DataMap. Set to false if it is an excel or csv file.
- string **Separator** [Get, Set]
  - The separator when reading a csv file. Default is “,”.
- string **Sheet** [Get, Set]
  - The name of the excel sheet to be read.
- string **StartCell** [Get, Set]
  - The name of the start cell to read the content from (for example, “B5”).
- string **StartCell** [Get, Set]
  - The name of the end cell to read the content from (for example, “H11”).
- bool **HasHeaders** [Get, Set]
  - Indicates whether or not the first row of the table content is a header row.
- bool **ConvertToJson** [Get, Set]
  - Whether to convert the content of the file into a json format.
- string **DataMapTarget** [Get, set]
  - The name of the key to be used in the DataMap to send the content of the file to.
- void **ReloadFile**()
  - Forces to read the content of the file. Might be used when **WatchFile** is disabled.

Copyright © 2025 Vizrt

Asset

- string **Value** [Get, Set]
  - Gets/sets the selected asset's fullpath.
  - Set: The input path needs to be valid.
- string **Prefix** [Get, Set]
  - The prefix to be added to the **Value** when sent to the engine.
- string **Postfix** [Get, Set]
  - The postfix to be added to the **Value** when sent to the engine.
- async void **SetImage** (string name)
  - An awaitable method to set an image *name*.

> ⚠ **Note:** Valid input values are: Graphic Hub items (Image, Geom, Material), Media service links (http://...) or local file system files.

WebView

This component lets you view a web page.

- string **Value** [Get, Set]
  - Gets/sets the URL of the web page to be visualized.
- void **Reload** ()
  - Reloads the current page.
- void **GoBack** ()
  - Navigates back in history.
- void **GoForward** ()
  - Navigates forward in history.
- void **ExecuteJavascript** (string method)
  - Invokes *method* on the currently loaded web page.
- void **ExecuteJavascript** (string method, params object[] args)
  - Invokes *method* with *args* as arguments on the currently loaded web page.

> ⚠ **Note:** The browser used for rendering is based on CEF. It might not play all video codecs.

A sample html file that might be loaded in a WebView:

```
<!DOCTYPE html>
<html>
<body>

<button onclick="DataMapButton()">Set Data Map</button>
<input type="text" id="someText" oninput="wroteSomeText()">

<p id="demo"></p>

<script>
function DataMapButton() {
```

Copyright © 2025 Vizrt

```
  let text = document.getElementById("someText").value;
  // set the data map variable 'fromWebPage' to the entered value
  arc.SetData("fromWebPage", text);
}
function wroteSomeText() {
  let text = document.getElementById("someText").value;
  document.getElementById("demo").innerHTML = "You wrote: " + text;
  // interact with arc and set the parameter "text_0" to the value just entered
  arc.SetParameterValue("text_0", text);
}

function someFunction1()
{
   alert("You invoked someFunction1!")
}

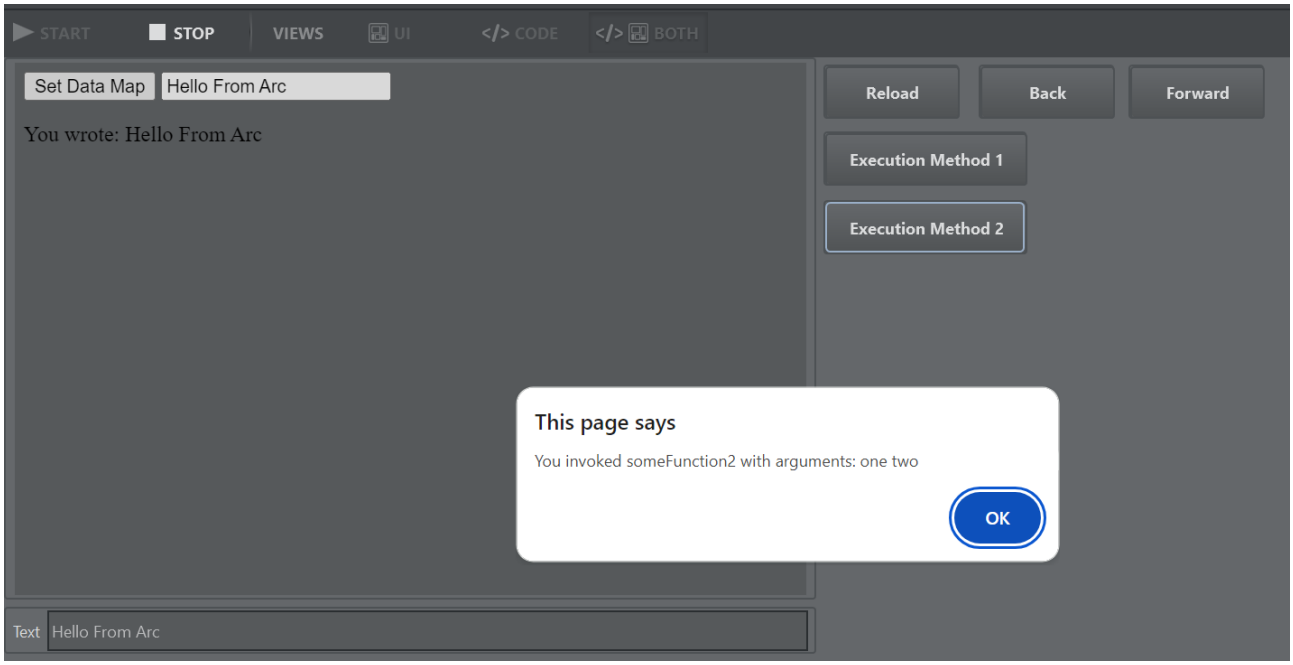
function someFunction2(para1, para2)
{
   alert("You invoked someFunction2 with arguments: " + para1 + " " + para2)
}
</script>

</body>

</html>
```

Note the **arc** object which gets injected and gives access to all scripting methods available in Viz Arc. For example *arc.SetData("matchResult", "0:1")* sets a variable on the **DataMap**.

A Viz Arc template interacting with the html page above after clicking the *Execute Method 2* button:



The code of the Viz Arc template might look like this, note the **ExecuteJavascript** methods that allow you to interact with the web page.

Copyright © 2025 Vizrt

```
Global.OnButtonPressed = function (id)
{
    switch( id )
    {
    case "reloadBrowser":
        webview_0.Reload()
        break
    case "backBrowser":
        webview_0.GoBack()
        break
    case "backBrowser":
        webview_0.GoFoward()
        break
    case "exe1":
        webview_0.ExecuteJavascript("someFunction1")
        break
    case "exe2":
        webview_0.ExecuteJavascript("someFunction2", "one", "two")
        break
    }
}
```

Bool

- bool **Value** [Get, Set]
    - Gets/sets the parameter's bool value.

Button

- void **Click** ( )
    - Trigger a click event on the button parameter.
- string **BackgroundImage**
    - Gets/sets the path to the background image. It can be either a local file path (for example, *c:/tmp/someimage.png*) a Graphic Hub path (for example, *IMAGE*/project/A/imageButton*) or a URL (for example, *http://storage.internal/image.jpg*).
- string **ImageMargin**
    - Gets/sets the margins in pixels of the background image. Specify either one, two or four comma separated margins. A value of `5` applies a margin of 5 pixels in all directions, a value of `5, 3` applies a margin of 5 pixels on the left and right and a margin of 3 at the top and bottom. A value of `1, 2, 3, 4` applies the respective margins in the order left, top, right and bottom.
- string **DirectorPath**
    - Gets/sets the Stage Director to be executed on click. For example, *$Director$SubDirector*
- string **DirectorExecute**
    - Gets/sets the type of action type to execute on click. Possible values are *<nothing>, START, CONTINUE, CONTINUE REVERSE, PAUSE or RESET*.
- string **ActionExecute**
    - Gets/sets the name or UUID of the action to be executed on click.

Copyright © 2025 Vizrt

Toggle Button

- void **SetCheckedColor** (byte r, byte g, byte b, byte a = 255)
  - Sets the toggle's color to the input RGBA color when the toggle is in it's checked state.
- string **CheckedColor**
  - Gets/sets the toggle's color in Hex format, for example, #FF0A0A8C (#RRGGBBAA) when the toggle is in it's checked state.
- int **CheckedColorR**
  - Gets/sets the toggle's red color value in the range **[0, 255]** when the toggle is in it's checked state.
- int **CheckedColorG**
  - Gets/sets the toggle's green color value in the range **[0, 255]** when the toggle is in it's checked state.
- int **CheckedColorB**
  - Gets/sets the toggle's blue color value in the range **[0, 255]** when the toggle is in it's checked state.
- int **CheckedColorA**
  - Gets/sets the toggle's alpha value in the range **[0, 255]** when the toggle is in it's checked state.
- string **BackgroundImage**
  - Gets/sets the path to the background image. It can be either a local file path (for example, *c:/tmp/someimage.png*) a Graphic Hub path (for example, *IMAGE*/project/A/imageButton*) or a URL (for example, *http://storage.internal/image.jpg*).
- string **ImageMargin**
  - Gets/sets the margins in pixels of the background image. Specify either one, two or four comma separated margins. A value of `5` applies a margin of 5 pixels in all directions, a value of `5, 3` applies a margin of 5 pixels on the left and right and a margin of 3 at the top and bottom. A value of `1, 2, 3, 4` applies the respective margins in the order left, top, right and bottom.
- bool **IsChecked**
  - Gets/sets the toggle button's state to checked or unchecked.
- string **ContainerPath**
  - Gets/sets the Viz scene container path (for example, *$object$ALL$left*) to be used when **VisibilityCheckd/VisibilityUncheckd** or **KeyChecked/KeyUnchecked** actions are set.
- string **DirectorPath**
  - Gets/sets the Stage Director path (for example, *$Director$SubDirector*)to be executed when **DirectorCheckd/DirectorUnchecked** actions are set.
- string **ActionChecked**
  - The action name or uuid to be executed when the toggle button gets **checked**.
- string **ActionUnchecked**
  - The action name or uuid to be executed when the toggle button gets **unchecked**.
- string **VisibilityChecked**
  - Gets/sets the visibility of the container specified in **ContainerPath** when the toggle button gets **checked**. Possible values are *<nothing>, ON or OFF*.
- string **VisibilityUnchecked**
  - Gets/sets the visibility of the container specified in **ContainerPath** when the toggle button gets **unchecked**. Possible values are *<nothing>, ON or OFF*
- string **DirectorChecked**
  - Gets/sets the action to be executed on the director specified in **DirectorPath** when the toggle gets **checked**. Possible values are *<nothing>, START, CONTINUE, CONTINUE REVERSE, PAUSE or RESET*.

Copyright © 2025 Vizrt

- string **DirectorUnchecked**
    - Gets/sets the action to be executed on the director specified in **DirectorPath** when the toggle gets **unchecked**. Possible values are *<nothing>, START, CONTINUE, CONTINUE REVERSE, PAUSE or RESET*.
- string **KeyChecked**
    - Gets/sets the key action of the container specified in **ContainerPath** when the toggle button gets **checked**. Possible values are *<nothing>, ACTIVE, INACTIVE, COMBINE WITH BG ON, COMBINE WITH BG OFF*.
- string **KeyUnchecked**
    - Gets/sets the key action of the container specified in **ContainerPath** when the toggle button gets **un checked**. Possible values are *<nothing>, ACTIVE, INACTIVE, COMBINE WITH BG ON, COMBINE WITH BG OFF*.

Double / Double Slider

- double **Value** [Get, Set]
    - Gets/sets the parameter's double value.
- double **MinValue** [Get, Set]
    - Gets/sets the parameter's minimum double value. Input value needs to be lower than the current MaxValue.
- double **MaxValue** [Get, Set]
    - Gets/sets the parameter's maximum double value. Input value needs to be higher than the current MinValue.
- double **MinRange** [Get, Set]
    - Gets/sets the parameter's minimum range double value.
- double **MaxRange** [Get, Set]
    - Gets/sets the parameter's maximum range double value.
- bool **RangeEnabled** [Get, Set]
    - If true, forces the slider to remain within the specified Min/Max range.
- bool **ShowReset** [Get, Set]
    - Enable or disable the reset to the default button on the UI.

Dropdown / Radio

- string **Value** [Get, Set]
    - Gets/sets the selected entry on the dropdown.
- int **SelectedIndex** [Get, Set]
    - Gets/sets the selected index of the dropdown.
- int **Count** [Get]
    - Gets the number of entries on the dropdown.
- int **IndexOf** (string option)
    - Looks for an entry equal to *option*. Returns its index if found, -1 otherwise.
- void **Insert** (int index, string option)
    - Inserts an entry with value *option* at *index* position. *index* needs to be between 0 and Count.
- void **Add** (string option)
    - Adds an entry with value *option* atthe end of the entry list.
- void **Remove** (string option)

Copyright © 2025 Vizrt

    - Looks for an entry equal to *option*. Removes it if found.
- void **RemoveAt** (int index)
    - Removes the entry at position *index*. *index* needs to be between 0 and Count.
- void **SetItems** (string[] entries)
    - Sets the dropdown's entry list to the input *entries*.
- string **Get** (int index)
    - Returns the entry located at *index* position. *index* needs to be between 0 and Count.
- string parameter[int index]
    - Array accessor for entries. Returns the entry located at *index* position.
- void **Clear** ( )
    - Removes all entries from the dropdown.

Int / Int Slider

- int **Value** [Get, Set]
    - Gets/sets the parameter's int value.
- int **MinValue** [Get, Set]
    - Gets/sets the parameter's minimum int value. Input value needs to be lower than the current MaxValue.
- int **MaxValue** [Get, Set]
    - Gets/sets the parameter's maximum int value. Input value needs to be higher than the current MinValue.

MultiText / Text

- string **Value** [Get, Set]
    - Gets/sets the parameter's text value.

Triplet

- double **X** [Get, Set]
    - Gets/sets the parameter's X double value.
- double **Y** [Get, Set]
    - Gets/sets the parameter's Y double value.
- double **Z** [Get, Set]
    - Gets/sets the parameter's Z double value.
- double **DefaultX** [Get, Set]
    - Gets/sets the default parameter’s X value.
- double **DefaultY** [Get, Set]
    - Gets/sets the default parameter’s Y value.
- double **DefaultZ** [Get, Set]
    - Gets/sets the default parameter’s Z value.
- bool **XEnabled** [Get, Set]
    - Gets/sets the enabled status of the X value.
- bool **YEnabled** [Get, Set]
    - Gets/sets the enabled status of the Y value.
- bool **Z Enabled** [Get, Set]

Copyright © 2025 Vizrt

    - Gets/sets the enabled status of the Z value.
- bool **AllowProportional** [Get, Set]
    - Gets/sets whether the user can toggle the proportional lock.
- bool **IsProportional** [Get, Set]
    - Gets/sets the state of the proportional lock.

Table

Properties

- string **Value** [Get]
    - Gets an string containing the table content in a XML format (much like ControlList).
- int **MinimumRows** [Get, Set]
    - Gets/sets the parameter's minimum number of rows. Input value needs to be lower than the current MaximumRows.
- int **MaximumRows** [Get, Set]
    - Gets/sets the parameter's maximum number of rows. Input value needs to be higher than the current MinimumRows.
- int **MinimumColumns** [Get, Set]
    - Gets/sets the parameter's minimum number of columns. Input value needs to be lower than the current MaximumColumns.
- int **MaximumColumns** [Get, Set]
    - Gets/sets the parameter's maximum number of columns. Input value needs to be higher than the current MinimumColumns.
- int **RowCount** [Get]
    - Gets the current number of rows on the table.
- int **ColumnCount** [Get]
    - Gets the current number of columns on the table.
- int **SelectedRow** [Get]
    - Gets the currently selected row. In case of multi-selection it returns the first selected row. If no row is selected, `-1` is returned.
- int **SelectedIndex** [Get]
    - An alias for **SelectedRow**.
- bool **MatchHeaders** [Get, Set]
    - Try to match headers of the incoming data, with the names of the column header names of the table.
- bool **MatchDataColumns** [Get, Set]
    - Create and match columns of the table according to incoming data.
- bool **MatchDataRows** [Get, Set]
    - Create and match as many rows in the table as there are in the incoming data.
- bool **TransposeData** [Get, Set]
    - Set to true to swap rows and columns.
- bool **TriggerOnAllChanges** [Get, Set]
    - Set to true to trigger the **OnTableCellValueChanged** callback, when a table cell value has been changed by user input.

Copyright © 2025 Vizrt

Methods

Cell Handling

- BaseCell **Accessor** [int row, int column] [Get]
  - Gets the cell located at *row*-indexed row and *column*-indexed column.
- BaseCell **GetCell** (int row, int column)
  - Gets the cell located at *row*-indexed row and *column*-indexed column.
- void **SetCellValue** (int row, int col, dynamic value)
  - Sets the cell's value (located at [row, column]) to *value.* [dynamic] *value* can either be a string or have a type that is compatible with the target cell.
- string**GetCellValue** (int row, int col)
  - Gets the cell's (located at [row, column]) string value representation.
- void **ClearColumnValues** (int columnIndex)
  - Resets all the cell's values in *columnIndex* column.
- void **ClearRowValues** (int rowIndex)
  - Resets all the cell's values in *rowIndex* row.
- void **ClearAllValues** ( )
  - Resets all the cell's values.
- void **Clear** ( )
  - Removes all the content (all columns and rows are deleted).

Columns Handling

Inserting a column from code requires the user to specify the type of column that needs to be created, the valid column types are:

- **bool:** Column with BoolCell
- **string:** Column with StringCell
- **int:** Column with IntCell
- **ivec2:** Column with IntDupletCell
- **ivec3:** Column with IntTripletCell
- **double:** Column with DoubleCell
- **dvec2:** Column with DoubleDupletCell
- **dvec3:** Column with DoubleTripletCell
- **asset:** Column with AssetCell

All column interactions take into consideration the maximum and minimum number of columns of the table

- void **AddColumn** (string columnType)
- void **AddColumn** (string columnType, string name)
  - Adds a column of type *columnType* named *name* if specified, otherwise the default is used.
- void **AddMultipleColumn** (int count, string columnType)
  - Adds *count* columns of *columnType* type.
- void **InsertColumn** (int index, string columnType)
- void **InsertColumn** (int index, string columnType, string name)
  - Inserts a column at *index* index of *columnType* type named *name* if specified, otherwise the default is used.
- vodi **InsertMultipleColumn** (int index, string columnType, int count)

Copyright © 2025 Vizrt

    - Inserts *count* columns at *index* index of *columnType* type.
- void **RemoveColumnAt** (int index)
    - Removes column at *index* index.
- void **MoveColumn** (int targetIndex, int newPosition)
    - Moves column from *targetIndex* position to *newPosition*.
- void **ClearColumns** ( )
    - Removes all columns.
- double **GetColumnWidth** (int index)
    - Returns the column *width* in pixels of column.
- void **SetColumnWidth** (int index, double width)
    - Sets the column *width* in pixels of column with index *index*.
- double **GetColumnName** (int index)
    - Returns the column *label*.
- void **SetColumnName** (int index, string name)
    - Sets the column label to *name* of column with index *index*.

Rows Handling

All row interactions take into consideration the maximum and minimum number of rows of the table

- void **SetNumberRows** (int count)
    - Adds/removes rows until the table's RowCount is equal to *count*.
- void **AddRow** ( )
    - Adds a Row to the table.
- void **AddMultipleRow** (int count)
    - Adds *count* rows to the table.
- void **InsertRow** (int index)
    - Inserts a row at *index* position to the table.
- void **InsertMultipleRow** (int index, int count)
    - Inserts *count* rows at *index* position to the table.
- void **RemoveRowAt** (int index)
    - Removes row at *index* position.
- void **MoveRow** (int targetIndex, int newPosition)
    - Moves row from *targetIndex* position to *newPosition*.
- void **ClearRows** ( )
    - Removes all rows.

Table Parameter Example

```
// Open the cvs file with the starters, parse the content and add all riders to the
RaceTable (TableParameter)
function LoadRaceTable()
{
    // Setup columns from UI, Comment if already done manually
    //RaceTable.Clear()
    //RaceTable.AddColumn( "string", "Horse")
    //RaceTable.AddColumn( "string", "Trainer")
```

Copyright © 2025 Vizrt

```
    //RaceTable.AddColumn( "string", "Jockey")
    //RaceTable.AddColumn( "string", "Owner")
    //RaceTable.AddColumn( "string", "Colors")
    //RaceTable.AddColumn( "string", "Horse CN")
    //RaceTable.AddColumn( "asset", "Silk")

    // Clear rows
    RaceTable.ClearRows()

    var i = 0
    var FileContent = arc.ReadTextFile("D:/Horses/Starter.csv")
    var EntryArr = FileContent.split("\n")

    // First line is for the headers, ignore it
    for(i = 1; i < EntryArr.length; i++)
    {
        // Split the rider content
        var splitContent = EntryArr[i].split(",")

        // CVS file has great amount of data but we only want to display certain
stuff
        RaceTable.AddRow()
        RaceTable.GetCell(i-1 , 0).Value = splitContent [19]
        RaceTable.GetCell(i-1 , 1).Value = splitContent [22]
        RaceTable.GetCell(i-1 , 2).Value = splitContent [25]
        RaceTable.GetCell(i-1 , 3).Value = splitContent [27]
        RaceTable.GetCell(i-1 , 4).Value = splitContent [34]
        RaceTable.GetCell(i-1 , 5).Value = splitContent [20]
        // image assets need to be assigned using the SetCellValue method
        RaceTable.SetCellValue(i-1 , 6, splitContent[21])
    }
}
```

Unreal

These are helper functions to invoke BluePrint functions:

- void **InvokeBPFunction** (string blueprintName, string functionName, params object[] arg)
    - Invokes the function *functionName* on the Blueprint *blueprintName,* with a variable number of parameters (typically strings and/or numbers).
- void **InvokeBPFunction** (string blueprintName, string functionName)
    - Invokes the function *functionName* on the Blueprint *blueprintName*, without any arguments.

The methods are invoked on all Unreal Engines of the template's currently selected channel.

```
Global.OnExecute = function ()
{
    // before executing the template, update some data on the Blueprint invoking
updateData
    InvokeBPFunction("BP_main", "updateData", "Hello", "World", 42.0, 3)
    // change the appearance of the car
```

Copyright © 2025 Vizrt

```
    InvokeBPFunction("BP_CarManager_Blue", "Change Car", 1, true, "This is the new
car model");
}
```

Video

Set the video and PTZ source.

- void **SetVideoSource** (string name)
    - Sets the name of the preview source window.
- void **SetNDIPTZVideoControl** (string ptzControl)
    - Sets the name of the NDI PTZ control overlay source.
- void **SetFlowicsOutput** (string URL)
    - Sets the global Flowics output URL.

Using async/await

Some methods are declared **async**, meaning the method might be time consuming (for example, while waiting for an answer from a server). To avoid locking up the UI, one can **await** an async method, such that Viz Arc can continue to process other events like user interaction. The async method might be processed on a different thread.

The following example shows how to retrieve the version of the first engine of the active channel on a template:

```
Global.OnButtonPressed = async function (id)
{
  if( id == "getFromEngineButton"){
    let answer = await GetFromEngineAsync("VERSION", 3000)
    Console.WriteLine("engine version is " + answer)
  }
}
```

Note that it is mandatory to declare the calling functions as **async,** when using **await**. The async keyword can be safely added to any Viz Arc callback, as shown with *OnButtonPressed*.

Using try/catch

It is highly recommended to use a try/catch block around awaited methods, otherwise, eventual errors are not detected.

```
Global.OnButtonPressed = async function (id)
{
  try{
    if( id == "getFromEngineButton"){
      let answer = await GetFromEngineAsync("VERSION", 3000)
      Console.WriteLine("engine version is " + answer)
    }
  }catch(ex)
  {
    Console.WriteLine("Error " + ex + " " + ex.stack)
  }
```

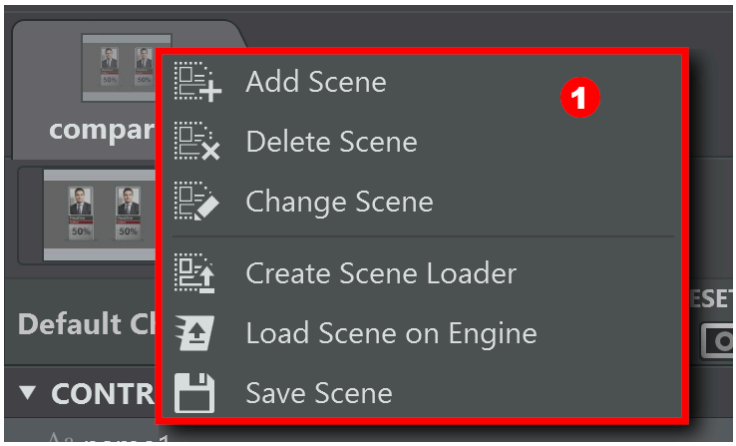
Copyright © 2025 Vizrt

```
}
```

5.5.4 Scenes Panel

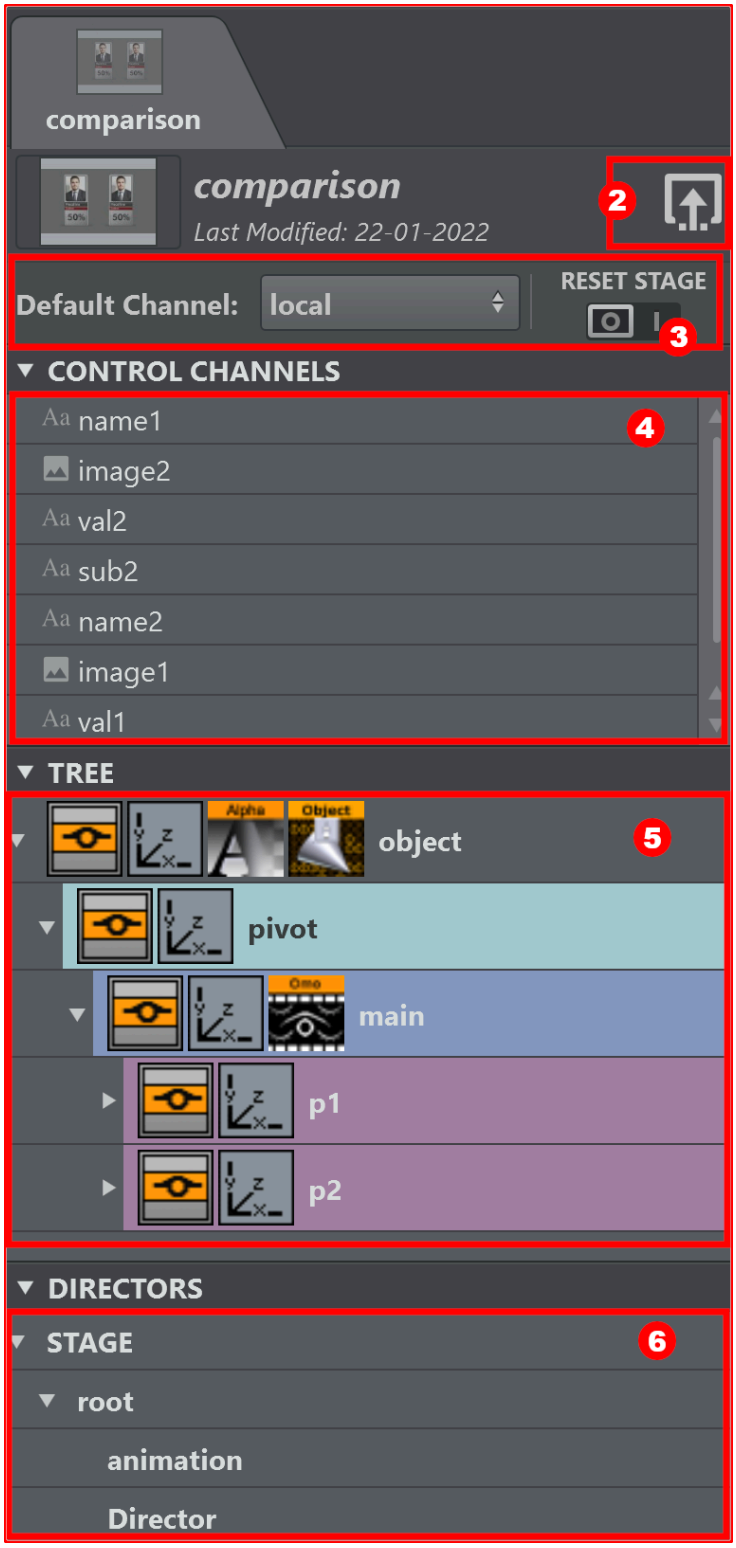
In Viz Arc, actions control different parameters within the Vizrt scene. The scenes panel is available in Actions and Set when working in Builder mode.

Right clicking the scene tabs or in the scene panel always gives you a context menu:



The Scenes Panel contains the following sections:

Copyright © 2025 Vizrt



1. Scene Panel
2. Load Scene
3. Scene Settings
4. Control Channels

Copyright © 2025 Vizrt

5. Scene Tree
6. Stage Directors

Managing Scenes

Managing the scenes in a Viz Arc project can be done from the scene details area of the Scenes Panel or from the window inside the Scenes Toolbar and existing scene loader.

This section covers the following topics:

- Add a Scene
- Change or Replace a Scene
- Delete a Scene

The Scene Panel

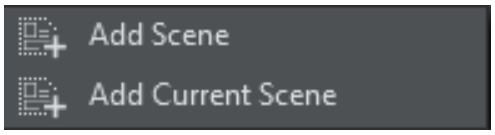
There are different ways to add a scene to a project:

When No Scene Is Added to a Project



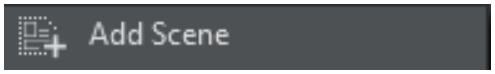
1. **Click** the the **ADD SCENE** button in the middle of the scene panel or inside the Scenes Toolbar window.
2. **Browse** and **double-click** the desired scene in the scenes browser.

Use the Context Menu on the Scenes Panel



Right-click on the top area of the Scenes Panel.

1. Select **Add Scene** from the context menu.
    a. Browse and double click the desired scene in the scenes browser.
2. Select **Add Current Scene** to add the currently loaded scene on the editing engine's main layer

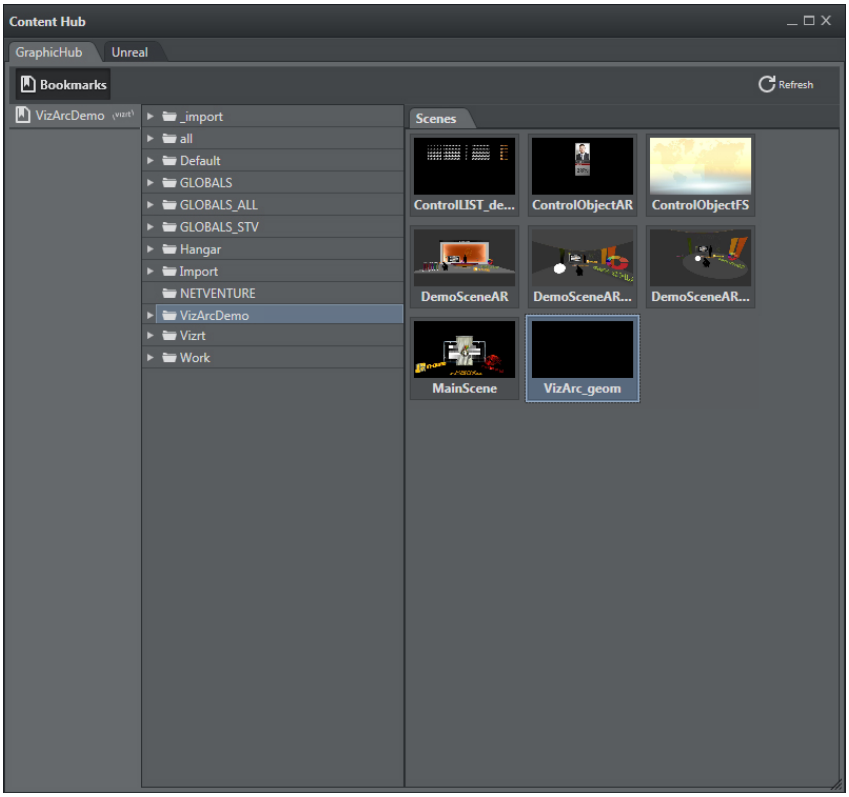


1. Click the **ADD SCENE** button at the top of the window inside the Scenes Toolbar.
2. Select **Add Scene** from the context menu.
3. Browse and double-click the desired scene.

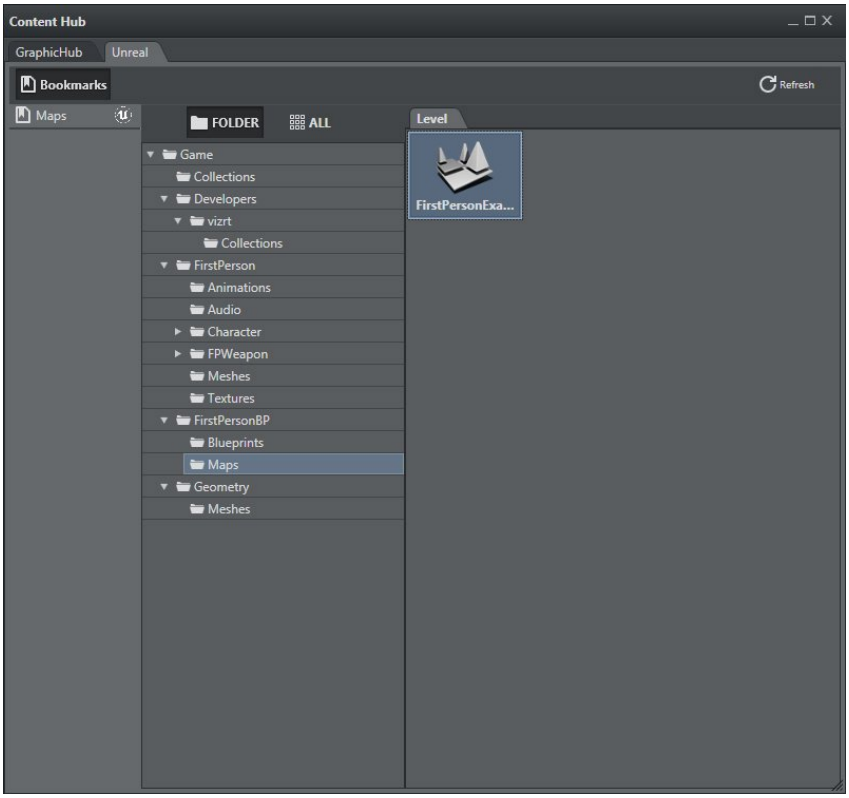
This opens the **Content Hub** window, where you can choose between Vizrt GH scenes or Unreal levels:

Vizrt

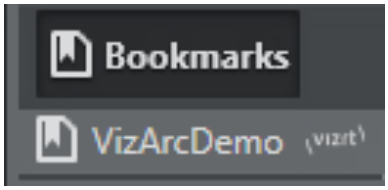
Copyright © 2025 Vizrt



Unreal Engine

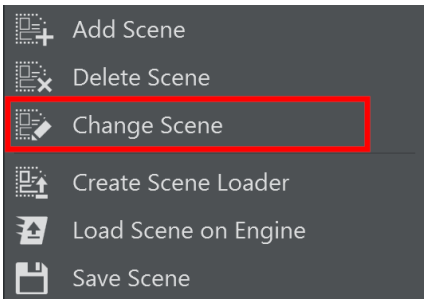


It's also possible to bookmark a specific folder in both the Vizrt and Unreal section:



Change or Replace a Scene

After a scene has been added to a project, it can be changed and replaced with a different scene. There are three ways to open the scenes browser to select a new scene:



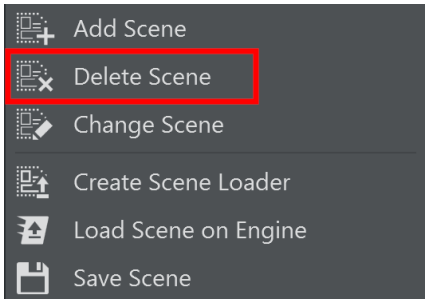
Right-click on the Scene tab from the scene panel, or

Right-click on the scene loader from the window inside the Scenes Toolbar.

Copyright © 2025 Vizrt

	**Click** the icon next to the scene name.
	**Click** the scene's thumbnail.

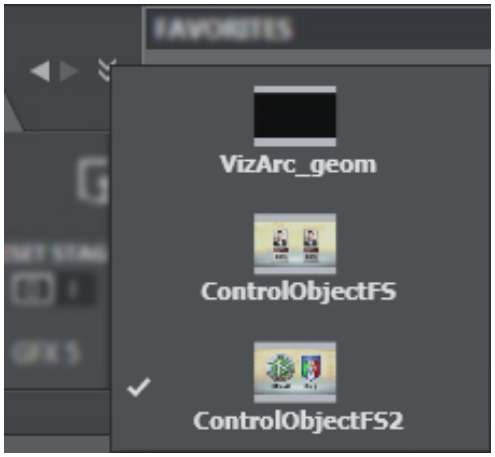
Delete a Scene



To delete a scene from a project:

1. **Right-click** the Scene tab from the scene panel, or
   Right Click the scene loader from the window inside the Scenes Toolbar.
2. Select **Delete Scene** from the context menu.
3. Select **Yes** to confirm.

Info: Note that the scene is NOT be deleted from Graphic Hub, only from the Scene Tabs in Viz Arc.

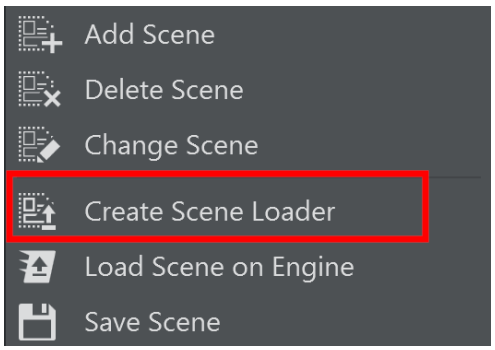


If there are more scenes in your Viz Arc project you can easily switch between them:

- using the drop-down scene menu, by selecting this icon
- or by clicking the left / right arrows

Copyright © 2025 Vizrt

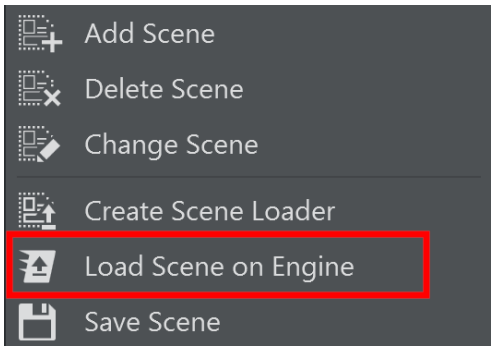
Create Scene Loader



To create a scene loader:

1. **Right-click** the Scene tab from the scene panel, or
   Right Click the scene loader from the window inside the Scenes Toolbar.
2. Select **Create Scene Loader** from the context menu.

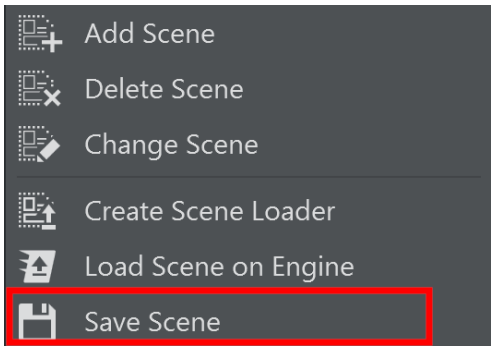
Load Scene on Engine



To load the scene on the Editing Engine:

1. **Right-click** the Scene tab from the scene panel, or
   Right Click the scene loader from the window inside the Scenes Toolbar.
2. Select **Load Scene on Engine** from the context menu.
   This will load the engine on the scene's **Default Channel**

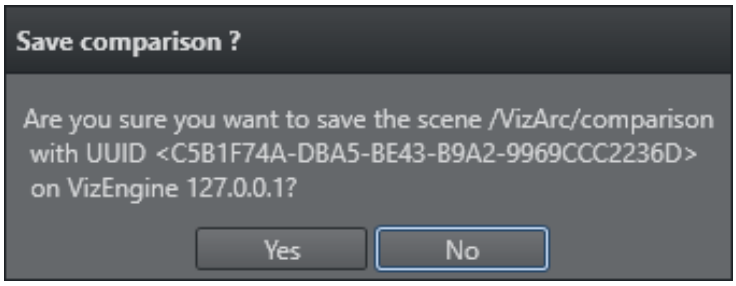
Save Scene



Copyright © 2025 Vizrt

To save the scene:

1. **Right-click** the Scene tab from the scene panel, or
   Right Click the scene loader from the window inside the Scenes Toolbar.
2. Select **Save Scene** from the context menu
3. Save the scene on the editing engine by clicking **Yes**:



Loading Scenes

Scenes **must** be loaded to a Viz Engine for Viz Arc to be able to retrieve all the scene properties (Control Channels, Scene Tree, Directors) and create actions in your project.

> **Note:** Before loading a scene, select a default channel and layer. For more details on selecting a default channel, see Scene Settings.

Scenes can be loaded in the following three ways:

	**Click** the **LOAD SCENE** button in the middle of the scene panel.
	**Click** the **Load** button in the scene details of the selected tab in the scene panel.

The scene will be loaded on the **Editing Engine** and will not be visible on the output. Use the context menu's **Load Scene on Engine** entry to load the scene on the **Editing Engine's** MAIN layer.

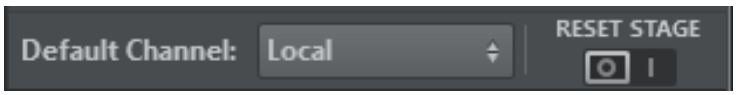
> Loading the scene on the **Editing Engine** might impact the engine's performance. Make sure the engine is not used on-air when parsing the scene.

Scene Settings

This section describes the settings that define what happens when loading scenes.

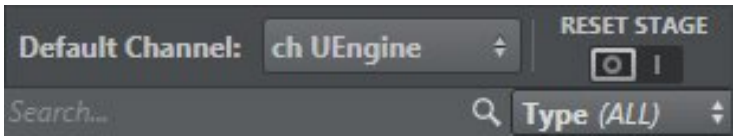
Copyright © 2025 Vizrt

Vizrt Scene Settings



- **Default Channel:** Uses the default channel to ease the live operation. All actions created from the scene properties are automatically assigned the same channel.
- **Reset Stage:** Resets all animations to frame `0` when loading the level, if enabled.

Unreal Level Settings



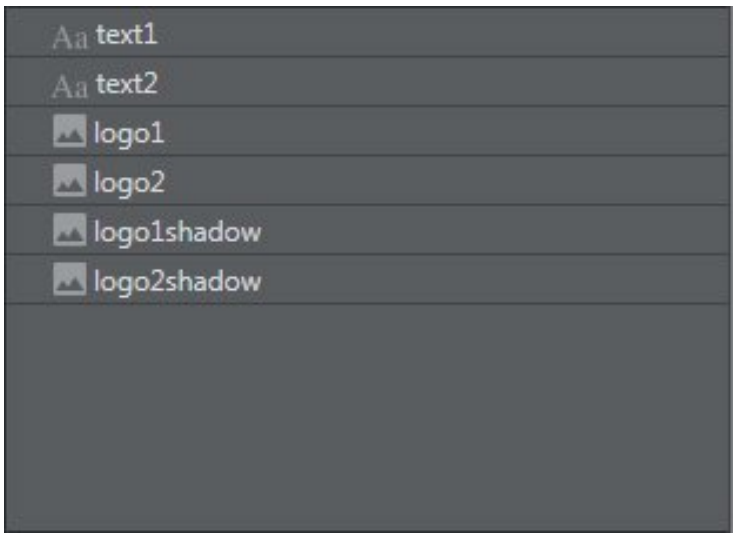
- **Default Channel:** Uses the default channel to ease the live operation. All actions created from the level properties are automatically assigned the same channel.
- **Reset Stage:** Resets all animations to frame `0` when loading the level, if enabled.
- **Search and Filter bar:** Searches for a specific element in the Level tree and filters actions among those available.

> **Note:** For more details on how to configure channels, see Profiles.

Control Channels

Viz Arc primarily connects an action to a scene property through **Control Channels**.

This section of the Scene Panel contains a list of all control channels available for the loaded Scene.



- Dragging **Control Channels** to the actions canvas is the *recommended* method for creating actions since it ensures actions do not break if the scene tree structure changes.

Copyright © 2025 Vizrt

- Actions are created based on the **Type** of **Control Channel**, which is represented by an **icon**.

Note: For more details about Control Channels, see the Viz Artist User Guide.

Scene Tree

This section of the Scene Panel contains a representation of the scene tree structure of the loaded Scene, as well as parameters and plug-ins supported as Viz Arc actions.



Actions created by dragging a parameter or plugin from the scene tree are connected using the container path.

Example: `$ContainerName$SubContName$Sub2ContName`)

Warning: To ensure actions are executed on the correct container, make sure there are no duplicate container names in the scene hierarchy.

Stage Directors

This section of the Scene Panel contains a list of any directors and sub-directors contained in the loaded Scene.

Copyright © 2025 Vizrt



Director Actions can be created by dragging any director to the Actions Canvas.

Hovering the cursor over each stage director (or director group) shows three buttons:

- : Sends a start command to Viz Engine.
- : Sends a continue command to Viz Engine.
- : Resets the selected stage (SHOW 0.0).

See Also

- Scene Action Types
- Viz Artist User Guide
- Viz Pilot User Guide

Copyright © 2025 Vizrt

5.6 APIs

- Viz Arc In-App Web API
- REST API

Copyright © 2025 Vizrt

5.6.1 Viz Arc In-App Web API

Viz Arc provides a Web API that allows you to set DataMap variables, trigger specific actions or control the Playlist.

The samples below assume that Arc is running on *localhost* and the In-App Web API is configured to run on port `5004.` The In-App Web API port can be configured in the General Settings, under the Communication section. Please note that this In-App Web API is only accessible when the Viz Arc application is running.

A Swagger webapp that describes every API can be found at http://localhost:5004/swagger/.

This section contains the following examples:

- Status
- ActionCanvas
- ActionExecuter
- Arena
- DataMap
- Object Tracker
- Playlist
- PlaylistExecuter
- Projects
- Profiles
- Web API Access Control
- Web Application Sample Usage

Information: The samples below use curl command line tools to demonstrate usage and python to beautify the returned JSON output. Note that these tools are not required for the Web API to function correctly.

Status

The Status endpoint allows you to get basic information of the running Viz Arc instance, such as version and uptime.

GET api/v1/status

curl -s -X GET http://localhost:5004/api/v1/Status | python -m json.tool

```
{
    "version": "Viz Arc 1.8.0.2116",
    "license": "CORE",
    "uptime": "00:59:14.8406020"
}
```

Copyright © 2025 Vizrt

ActionCanvas

The ActionCanvas endpoint allows you to retrieve all actions of a project, including information about their position, color, id etc.. It also allows you to select an action specified by the *id* or *name*. An interactive demo page using this API can be found at *http://localhost:5004/*.

GET /api/v1/actioncanvas

This endpoints returns all actions of the currently loaded project. The property "actions" contains an array of all actions. The property *tabs* contains the information about the project tabs.

curl -s -X GET http://localhost:5004/api/v1/ActionCanvas | python -m json.tool


```
{
    "tabs": [
        {
            "name": "show",
            "isEditorOnly": false
        },
        {
            "name": "Startlist",
            "isEditorOnly": false
        }
    ],
    "selectedTabIndex": 0,
    "actions": [
        {
            "uuid": "a090e1c9-28ad-4c4f-976d-2727befe182d",
            "name": "Title Day 1",
            "type": "TEMPLATE",
            "actionType": null,
            "engineType": "VIZ",
            "xSector": 1,
            "ySector": 0,
            "widthSector": 1,
            "heightSector": 1,
            "tab": 0,
            "delay": 0,
            "actionColor": "#65686B",
            "textColor": "#DBDCDD"
        },
        {
            "uuid": "95c0bc26-fb63-4ffe-becf-bd42db7ae5f8",
            "name": "Startlist",
            "type": "TEMPLATE",
            "actionType": null,
            "engineType": "VIZ",
            "xSector": 0,
            "ySector": 1,
            "widthSector": 1,
```

```
            "heightSector": 1,
            "tab": 1,
            "delay": 0,
            "actionColor": "#65686B",
            "textColor": "#DBDCDD"
        }
    ],
    "project": "Skating"
}
```

POST /api/v1/actioncanvas/select/{id | name}

Selects an action specified by the *id* or *name*.

curl -s -X POST http://localhost:5004/api/v1/ActionCanvas/select/13ddcaa8-e52a-4357-85d6-5731f3b45ebd

```
200 OK
```

ActionExecuter

The ActionExecuter endpoint allows you to execute a specific action either by using its name or its ID.

GET /api/v1/actionexecuter/{id | name}

Executes the action specified by the *id* or *name*.

curl -s -X GET http://localhost:5004/api/v1/ActionExecuter/27cc7a3b-6f90-4d88-9101-a356a1668d83

```
200 OK
```

Using the ID variant is preferable as it uniquely identifies an action. Although the name might appear multiple times in your action collection, only the first action with the specified name is triggered.

GET /api/v1/actionexecuter/{id | name}/continue

Executes a "Continue" on a template action specified by the *name*.

curl -s -X GET http://localhost:5004/api/v1/ActionExecuter/27cc7a3b-6f90-4d88-9101-a356a1668d83/continue

```
200 OK
```

GET /api/v1/actionexecuter/{id | name}/out

Executes a "Out" on a template action specified by the *name*.

Copyright © 2025 Vizrt

curl -s -X GET http://localhost:5004/api/v1/ActionExecuter/27cc7a3b-6f90-4d88-9101-a356a1668d83/out

```
200 OK
```

GET /api/v1/actionexecuter/{id | name}/update

Executes a "Update" on a template action specified by the *name*.

curl -s -X GET http://localhost:5004/api/v1/ActionExecuter/27cc7a3b-6f90-4d88-9101-a356a1668d83/update

```
200 OK
```

Arena

The Arena APIs allow you to control the Viz Arena integration.

GET /api/v1/arena/detectArenaCalibration

Detects the calibration.

curl -s -X GET http://localhost:5004/api/v1/arena/detectArenaCalibration | python -m json.tool

```
200 OK
```

GET /api/v1/arena/clearArenaCalibration

Clears the current calibration.

curl -s -X GET http://localhost:5004/api/v1/arena/clearArenaCalibration | python -m json.tool

```
200 OK
```

GET /api/v1/arena/clearArenaKeyer

Clears the Viz Arena keyer.

curl -s -X GET http://localhost:5004/api/v1/arena/clearArenaKeyer | python -m json.tool

```
200 OK
```

Copyright © 2025 Vizrt

GET /api/v1/arena/setArenaCamera/{id}

Sets the active arena camera.

curl -s -X GET http://localhost:5004/api/v1/arena/setArenaCamera/1 | python -m json.tool

```
200 OK
```

DataMap

You can set, read and delete DataMap variables through the Web API. An interactive web application can be found at *http://localhost:5004/DataMap*.

GET /api/v1/DataMap

Gets the list of all the key-value pairs.

curl -s -X GET http://localhost:5004/api/v1/DataMap | python -m json.tool

```
[
    {
        "key": "Hello",
        "value": "World!"
    },
    {
        "key": "WhiteListIPRanges",
        "value": "0.0.0.0/0"
    }
]
```

GET /api/v1/DataMap/{key}

Gets the key-value pair identified by the specified key.

curl -s -X GET http://localhost:5004/api/v1/DataMap/key | python -m json.tool

```
{
    "key": "Hello",
    "value": "World!"
}
```

Copyright © 2025 Vizrt

POST /api/v1/DataMap

Adds a new key-value pair or an array of them to the DataMap. The payload must be sent as JSON body. If the key already exists the value is updated.

curl -s -H 'Content-Type: application/json' -d "{ 'Key': 'Hello', 'Value': 'World!' }" -X POST http://localhost:5004/api/v1/DataMap

```
{
    "Key": "key",
    "Value": "hello"
}
```

DELETE /api/v1/DataMap/{key}

Deletes a key/value pair by sending a DELETE message, for example to delete "key2" use:

curl -s -X DELETE http://localhost:5004/api/v1/DataMap/key2 | python -m json.tool

```
200 OK
```

Object Tracker

The Object Tracker APIs allow you to control the Object Tracker integration.

GET /api/v1/objectTracker/takeTracker

Takes all active trackers On Air.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/takeTracker | python -m json.tool

```
200 OK
```

GET /api/v1/objectTracker/takeOutTracker

Takes all active trackers Off Air.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/takeOutTracker | python -m json.tool

```
200 OK
```

Copyright © 2025 Vizrt

GET /api/v1/objectTracker/previewTracker

Previews all active trackers on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/PreviewTracker | python -m json.tool

```
200 OK
```

GET /api/v1/objectTracker/previewOutTracker

Previews out all active trackers on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/PreviewOutTracker | python -m json.tool

```
200 OK
```

GET /api/v1/objectTracker/stopTracker/{id}

Stops the tracker with index {id}, starting from index 1.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/stopTracker/1 | python -m json.tool

```
200 OK
```

GET /api/v1/objectTracker/setActiveTracker/{id}

Sets the active tracker with index {id}, starting from index 1.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/setActiveTracker/1 | python -m json.tool

```
200 OK
```

GET /api/v1/objectTracker/resetPointerOffset/{id}

Resets the pointer offset of the tracker with index {id}, starting from index 1.

curl -s -X GET http://localhost:5004/api/v1/objectTracker/resetPointerOffset/1 | python -m json.tool

```
200 OK
```

Copyright © 2025 Vizrt

Playlist

You can read, trigger, preview and iterate through the playlist through the Web API. An interactive web application can be found at *http://localhost:5004/PlayList*.

GET /api/v1/playlist

Gets all the tabs.

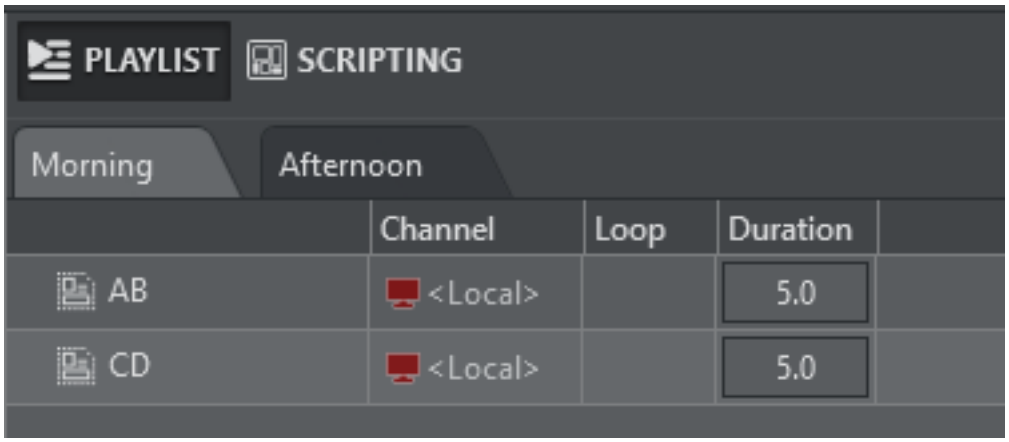
curl -s -X GET http://localhost:5004/api/v1/PlayList | python -m json.tool


```
[
    {
        "duration": 10.0,
        "index": 0,
        "name": "Morning",
        "rows": [
            {
                "actionUUID": "b78801bd-c802-42dd-ae11-9955c53dbfe3",
                "duration": 5.0,
                "id": "aaf0b451-1bb6-4bae-b11d-3b0c5f552e36",
                "name": "AB"
            },
            {
                "actionUUID": "6dd3107a-9487-4f32-92a0-b7eb83a1328f",
                "duration": 5.0,
                "id": "58b51c71-a904-4c95-b5d8-f42184af025e",
                "name": "CD"
            }
        ]
    },
    {
        "duration": 15.0,
        "index": 1,
        "name": "Afternoon",
        "rows": [
            {
                "actionUUID": "27cc7a3b-6f90-4d88-9101-a356a1668d83",
                "duration": 5.0,
                "id": "7f284bd1-a727-4e26-aea8-f417010c5386",
                "name": "Opener"
            },
            {
                "actionUUID": "955a622a-862b-4206-9658-a48b0c9f9a7e",
                "duration": 5.0,
                "id": "f43b4470-d08c-4261-8fb2-58ef97c4db28",
                "name": "Closer"
            },
            {
                "duration": 5.0,
```

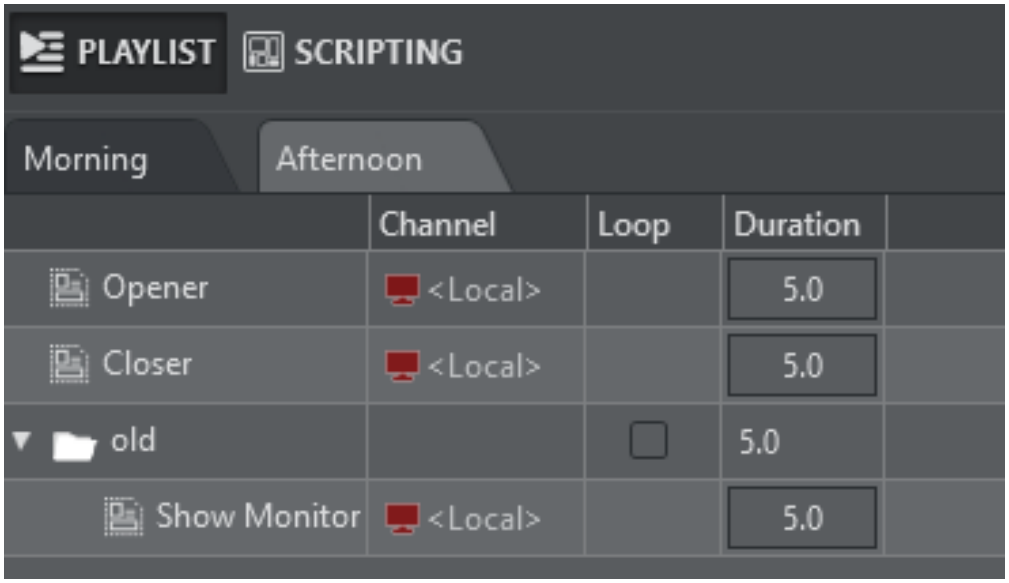

Copyright © 2025 Vizrt

```
                "id": "483231c0-5484-478c-9bbc-74530dafa7cd",
                "loop": false,
                "name": "old",
                "rows": [
                    {
                        "actionUUID": "ae7d2e75-552f-40e7-8c3d-7a8ef716809a",
                        "duration": 5.0,
                        "id": "c542f3d4-53d3-4aa1-82d1-8e75ae86342b",
                        "name": "Show Monitor"
                    }
                ]
            }
        ]
    }
]
```


The GET method returns all Playlist tabs and their respective rows and folders.



The screenshot above shows the query result for the "Morning" Playlist tab.



The screenshot above shows the query result for the "Afternoon" Playlist tab.

Copyright © 2025 Vizrt

GET /api/v1/playlist/{index}

Gets the content of the playlist tab, by specifying its index.

curl -s -X GET http://localhost:5004/api/v1/PlayList/0 | python -m json.tool


```
{
    "duration": 10.0,
    "index": 0,
    "name": "Morning",
    "rows": [
        {
            "actionUUID": "b78801bd-c802-42dd-ae11-9955c53dbfe3",
            "duration": 5.0,
            "id": "17a6ee8d-8c82-4480-9bda-32e4b9447336",
            "name": "AB"
        },
        {
            "actionUUID": "6dd3107a-9487-4f32-92a0-b7eb83a1328f",
            "duration": 5.0,
            "id": "4580c837-69a9-41b4-95dc-29b05f1ae0d4",
            "name": "CD"
        }
    ]
}
```


Starting from index 0 this call returns a single playlist tab only.

PlaylistExecuter

The PlaylistExecuter endpoint allow you to execute and preview individual playlist elements identified by their id.

GET api/v1/PlaylistExecuter/execute/{id}

Executes the playlist element identified by *id.*

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/execute/2628b63d-e947-4ee6-9dd2-5c90525e2cf5 | python -m json.tool

```
200 OK
```

GET api/v1/PlaylistExecuter/continue/{id}

Continues the playlist element identified by *id.*

Copyright © 2025 Vizrt

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/continue/2628b63d-e947-4ee6-9dd2-5c90525e2cf5 | python -m json.tool

```
200 OK
```

GET api/v1/PlaylistExecuter/preview/{id}

Executes the playlist element identified by *id* on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/preview/2628b63d-e947-4ee6-9dd2-5c90525e2cf5 | python -m json.tool

```
200 OK
```

GET api/v1/PlaylistExecuter/previewContinue/{id}

Continues the playlist element identified by *id* on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/previewContinue/2628b63d-e947-4ee6-9dd2-5c90525e2cf5 | python -m json.tool

```
200 OK
```

GET api/v1/PlaylistExecuter/playPlaylistByName/{name}

Plays every item in the playlist identified by *name*.

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/playPlaylistByName/myPlaylist | python -m json.tool

```
200 OK
```

GET api/v1/PlaylistExecuter/playPlaylistByIndex/{id}

Plays every item in the playlist identified by *Id*.

curl -s -X GET http://localhost:5004/api/v1/PlaylistExecuter/playPlaylistByIndex/1 | python -m json.tool

```
200 OK
```

Copyright © 2025 Vizrt

Playlist Controls on the Currently Selected Playlist

The following commands are invoked against the Playlist that is currently selected in Viz Arc:

GET api/v1/playlistexecuter/executeCurrent

Executes the currently selected playlist element.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/executeCurrent

```
200 OK
```

GET api/v1/playlistexecuter/continueCurrent

Continues the currently selected playlist element.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/continueCurrent | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/outCurrent

Takes out the currently selected playlist element (only for template actions).

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/outCurrent | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/previewCurrent

Executes the currently selected playlist element on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/previewCurrent | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/previewContinueCurrent

Continues the currently selected playlist element on the preview channel.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/previewContinueCurrent | python -m json.tool

```
200 OK
```

Copyright © 2025 Vizrt

GET api/v1/playlistexecuter/previewOutCurrent

Takes out the currently selected playlist element on the preview channel (only for template actions).

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/previewOutCurrent | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/executeAndNext

Executes the currently selected playlist element and moves to the next playlist item.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/executeAndNext | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/gotoFirst

Selects the first playlist item.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/gotoFirst | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/playCurrentPlaylist

Plays every item in the currently selected playlist.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/playCurrentPlaylist | python -m json.tool

```
200 OK
```

GET api/v1/playlistexecuter/stopCurrentPlaylist

Stops the execution of the currently selected playlist.

curl -s -X GET http://localhost:5004/api/v1/playlistexecuter/stopCurrentPlaylist | python -m json.tool

```
200 OK
```

Projects

The Projects endpoint allows you to get a list of all available projects and to load a project.

Copyright © 2025 Vizrt

Information: Viz Arc can store and read projects from two different locations, **Graphic Hub** or the local **file system**. In order to distinguish the source of the project, the project's name is prefixed by the keywords "*gh:*" or "*local:*" (for example, "gh:sunday_elections" or "*local:sunday_elections*").
Also, the "*location*" property in the responses, indicates whether the project resides on Graphic Hub (value "*1*") or the file system (value "*0*").

GET api/v1/projects

Returns a list of all available Projects, first the projects located on the Graphic Hub, then the ones on the file system.

curl -s -X GET http://localhost:5004/api/v1/Projects | python -m json.tool


```
[
    {
        "name": "gh:Animations",
        "lastModified": "2023-05-09T13:53:32.4633996+00:00",
        "location": 1,
        "profileIndex": 0,
        "tabs": []
    },
    {
        "name": "local:Skating",
        "lastModified": "2023-05-11T07:53:51.80198+00:00",
        "location": 0,
        "profileIndex": 0,
        "tabs": []
    }
]
```


In the example above, two projects were found.

The name of the project is prefixed by the keywords "gh:" or "local:"; this prefix indicates whether the project is stored in the Graphic Hub or on the file system.

GET api/v1/projects/{id}

Returns the project with the specified *id*, if it exists. If two projects with the same name exist in Graphic Hub and the local disk, the one of the local disk is returned first.

curl -s -X GET http://localhost:5004/api/v1/Projects/Animations | python -m json.tool


```
{
    "name": "gh:Animations",
    "lastModified": "2023-05-09T13:53:32.4633996+00:00",
    "location": 1,
    "profileIndex": 0,
    "tabs": []
}
```


Copyright © 2025 Vizrt

GET api/v1/projects/load/{id}

Loads the project with the specified *id* in Viz Arc. If the *id* does not contain the location prefix, then Viz Arc tries to retrieve the project with a given name by looking in the Graphic Hub first, then the file system.

curl -s -X GET http://localhost:5004/api/v1/Projects/load/local:Skating | python -m json.tool

```
200 OK
```

GET api/v1/projects/current

Gets the name and location (*Local* or *GraphicHub*) of the currently loaded project.

curl -s -X GET http://localhost:5004/api/v1/projects/current | python -m json.tool


```
{
    "name": "local:Skating",
    "lastModified": "2023-05-11T07:53:51.80198+00:00",
    "location": 0,
    "profileIndex": 2,
    "tabs": [
        {
            "name": "show",
            "index": 0,
            "isEditorOnly": false,
            "actions": [
                {
                    "uuid": "ff1ae771-4c69-42e4-ba8a-ad98747a5477",
                    "name": "Logo",
                    "type": "Template",
                    "actionType": "TEMPLATE",
                    "engineType": "VIZ",
                ...
```


Profiles

The Profiles endpoint allows you to retrieve information about the available Profiles, Channels and Engines.

GET /api/v1/profiles

Returns the entire list of profiles available in Viz Arc.

curl -s -X GET http://localhost:5004/api/v1/profiles | python -m json.tool


```
[
```


Copyright © 2025 Vizrt

```
    {
        "name": "Local",
        "channels": [
            {
                "name": "Local",
                "type": 1,
                "engines": [
                    {
                        "name": "localhost"
                    }
                ],
                "model": {
                    "engineConfigTuple": [
                        {
                            "item1": {
                                "name": "localhost",
                                "ipAddress": "127.0.0.1",
                                "port": 6100,
                                "launcherPort": 5644,
                                "webInterfacePort": 30010,
                                "webSocketPort": 30020,
                                ...
```


GET api/v1/profiles/{id | name}

Returns the profile specified by the *id* or *name*.

curl -s -X GET http://localhost:5004/api/v1/profiles/0 | python -m json.tool


```
{
    "name": "Local",
    "channels": [
        {
            "name": "Local",
            "type": 1,
            "engines": [
                {
                    "name": "localhost"
                }
            ],
            "model": {
                "engineConfigTuple": [
                    {
                        "item1": {
                            "name": "localhost",
                            "ipAddress": "127.0.0.1",
                            "port": 6100,
                            "launcherPort": 5644,
                            "webInterfacePort": 30010,
                            "webSocketPort": 30020,
```


Copyright © 2025 Vizrt

```
                                ...
```

GET api/v1/profiles/current

Returns the currently selected profile.

curl -s -X GET http://localhost:5004/api/v1/profiles/current | python -m json.tool

```
{
    "name": "Local",
    "channels": [
        {
            "name": "Local",
            "type": 1,
            "engines": [
                {
                    "name": "localhost"
                }
            ],
            "model": {
                "engineConfigTuple": [
                    {
                        "item1": {
                            "name": "localhost",
                            "ipAddress": "127.0.0.1",
                            "port": 6100,
                            "launcherPort": 5644,
                            "webInterfacePort": 30010,
                            "webSocketPort": 30020,
                            ...
```

POST api/v1/profiles

Sets the currently active profile.

curl -s -H 'Content-Type: application/json' -d "{ 'name': 'production' }" -X POST http://localhost:5004/api/v1/profiles

```
{
    "name": "productionA"
}
```

curl -s -H 'Content-Type: application/json' -d "{ 'index': 0 }" -X POST http://localhost:5004/api/v1/profiles

```
{
    "index": 0
```

Copyright © 2025 Vizrt

```
}
```

Index starts at 0 for the first profile.

Web API Access Control

It is possible to inhibit calls to the **PlayListExecuter** and the **ActionExecuter** APIs with the IP whitelisting.

IP Whitelist

You can configure the IP addresses/ranges that are allowed to use the APIs. These have to be specified using a semi-colon (**;**) separated list of IP ranges (in CIDR notation). By default, the range is 0.0.0.0/0 which means all hosts can access the API.

The IP Whitelist can be configured in three different ways:

- Manually in the General Settings(under the Communication section)
- Web Application
- Scripting

Web Application Whitelist Configuration

If you want to allow only two machines with IP addresses (e.g. *192.168.1.100* and *192.168.1.142)* you can specify them by reaching the page http://localhost:5004/home/settings#datamap and set the value of the **WhiteListIPRanges** key, as in the snapshot below.



Scripting Whitelist Configuration

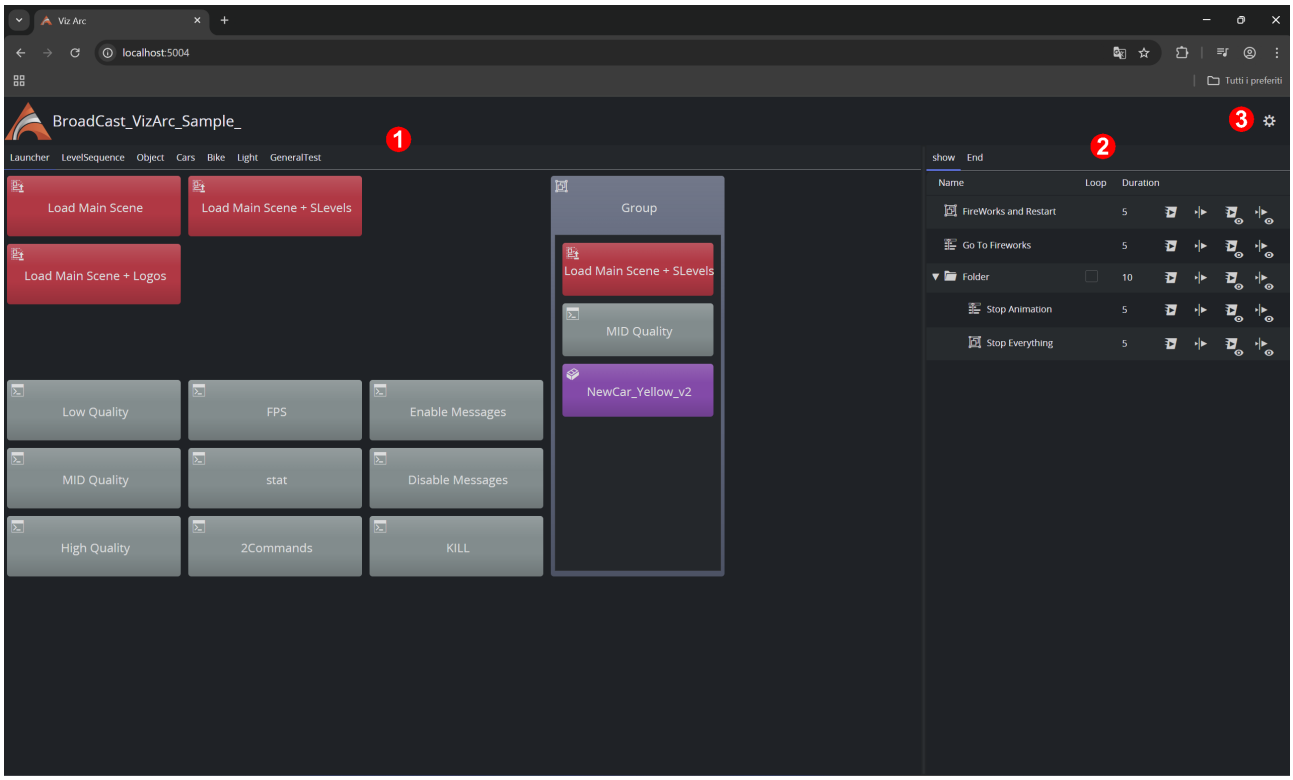
Set Whitelist IPs through Scripting

```
SetData("WhiteListIPRanges", "192.168.1.100;192.168.1.142")
```

Web Application Sample Usage

When Viz Arc starts up, it also launches an internal In-App Web Server which host a Web Application that makes use of some of the APIs described above.

Navigate to http://localhost:5004/ (the port can be configured in the General Settings, under the Communication section) to see the currently loaded project in Viz Arc. For example:



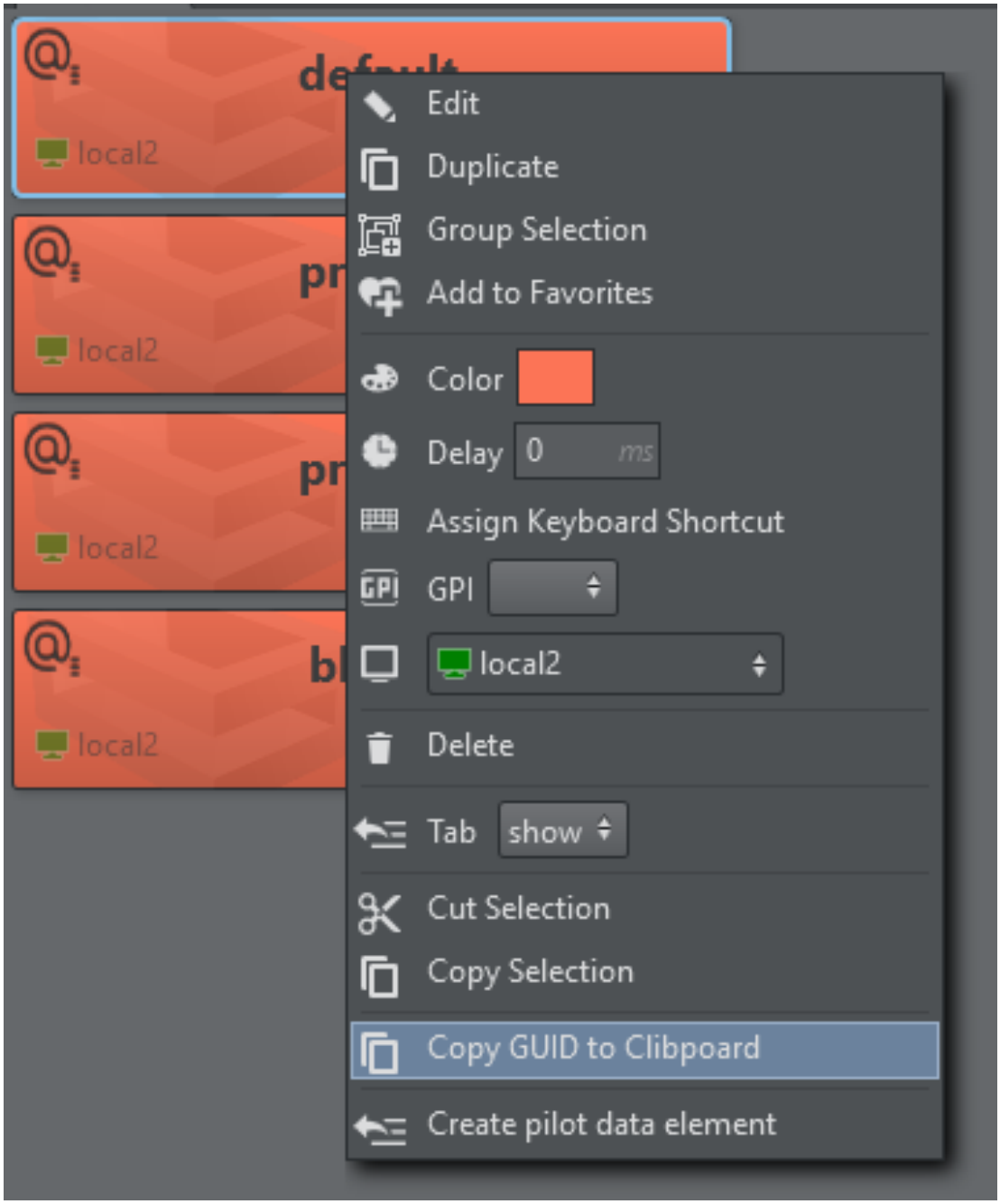
1. **Actions**: the project loaded in Arc. Click any action to execute it.
2. **PlayList**: trigger all available playlists.
3. **Datamap**: click to open a new dialog. From here you can add, delete or modify any DataMap variable.



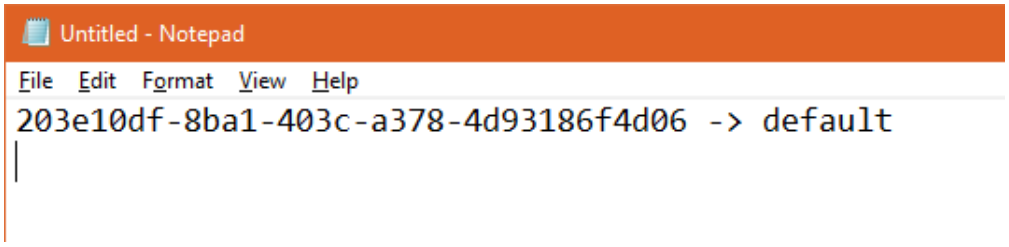
Triggering a Viz Arc Action from the Web Browser

You can easily trigger a Viz Arc action for the currently loaded project using its GUID or just its name. First, get the GUID and name from the action:

Copyright © 2025 Vizrt



The GUID is now copied to the clipboard from where you can easily paste it anywhere else (for example, to notepad):



Copyright © 2025 Vizrt

Typing *http://localhost:5004/api/v1/ActionExecuter/203e10df-8ba1-403c-a378-4d93186f4d06* *or* *http://localhost:5004/api/v1/ActionExecuter/default* triggers the action.

Information: Be aware that triggering only works when Viz Arc is running and when the project that contains the requested Action is open.

Copyright © 2025 Vizrt

5.6.2 REST API

Viz Arc Rest Service provides a Web API that allows you to trigger specific actions or control the Playlist.

The samples below assume that you have installed Viz Arc REST Service and that it's running on *localhost* and the REST Web API is configured to run on port `9004`. The REST Web API port can be configured in the General Settings, under the Communication section.

A Swagger webapp that describes every API can be found at *http://localhost:9004/swagger/*.

This section contains the following examples:

- DataMap
- Executer
- Projects
- Profiles
- Web Application Sample Usage
- Keyer

> **Info:** The samples below use curl command line tools to demonstrate usage and python to beautify the returned JSON output. Note that these tools are not required for the Web API to function correctly.

DataMap

You can set, read and delete DataMap variables through the Web API.

GET /api/v1/DataMap

Gets the list of all the key-value pairs.

curl -s -X GET http://localhost:9004/api/v1/DataMap | python -m json.tool

```
[
    {
        "key": "Hello",
        "value": "World!"
    },
    {
        "key": "WhiteListIPRanges",
        "value": "0.0.0.0/0"
    }
]
```

GET /api/v1/DataMap/{key}

Gets the key-value pair identified by the specified key.

Copyright © 2025 Vizrt

curl -s -X GET http://localhost:9004/api/v1/DataMap/key | python -m json.tool

```
{
    "key": "Hello",
    "value": "World!"
}
```

POST /api/v1/DataMap

Adds a new key-value pair or an array of them to the DataMap. The payload must be sent as JSON body. If the key already exists the value is updated.

curl -s -H 'Content-Type: application/json' -d "{ 'Key': 'Hello', 'Value': 'World!' }" -X POST http://localhost:9004/api/v1/DataMap

```
{
    "Key": "key",
    "Value": "hello"
}
```

DELETE /api/v1/DataMap/{key}

Delete a key/value pair by sending a DELETE message. For example, to delete "key2" use:

curl -s -X DELETE http://localhost:9004/api/v1/DataMap/key2 | python -m json.tool

```
200 OK
```

Executer

The Executer endpoints allows you to execute a specific action either by using its name or its ID.

POST api/v1/Executer

Executes the selected action. In order to identify a specific action, the user must specify the *actionId*, the *projectId*, the *profileId* and the *executionType.*

curl -s -H 'Content-Type: application/json' -d "{ 'actionId': 'a45dbfda-d197-49ff-9995-66d3c319e1fc', 'projectId': 'gh:Sunday_Elections', 'profileId': 'Profile-1', 'executionType': 'EXECUTE' }" -X POST http://localhost:9004/api/v1/Executer

```
200 OK
```

Copyright © 2025 Vizrt

Projects

The Projects endpoint allows you to get a list of all available projects and to load a project.

> **Info:** Viz Arc can store and read projects from two different locations: the **Graphic Hub** or the local **file system**. In order to distinguish the source of the project, the project's name is prefixed by the keywords "*gh:*" or "*local:*" (e.g. "gh:sunday_elections" or "*local:sunday_elections*").
> Likewise, the "*location*" property in the responses, indicates whether the project resides on Graphic Hub (value "*1*") or on the file system (value "*0*").

GET api/v1/projects

Returns a list of all available Projects, first the projects located on the Graphic Hub, then the ones on the file system.

curl -s -X GET http://localhost:9004/api/v1/Projects | python -m json.tool

```
[
    {
        "name": "gh:Animations",
        "lastModified": "2023-05-09T13:53:32.4633996+00:00",
        "location": 1,
        "profileIndex": 0,
        "tabs": []
    },
    {
        "name": "local:Skating",
        "lastModified": "2023-05-11T07:53:51.80198+00:00",
        "location": 0,
        "profileIndex": 0,
        "tabs": []
    }
]
```

In the example above, two projects were found.

The name of the project is prefixed by the keywords "gh:" or "local:"; this prefix indicates whether the project is stored in the Graphic Hub or on the file system.

GET api/v1/projects/{id}

Returns the project with the specified *id*, if it exists. If two projects with the same name exist in the GH and the local disk, the one of the local disk will be returned first.

curl -s -X GET http://localhost:9004/api/v1/Projects/Animations | python -m json.tool

```
{
    "name": "gh:Animations",
    "lastModified": "2023-05-09T13:53:32.4633996+00:00",
```

Copyright © 2025 Vizrt

```
    "location": 1,
    "profileIndex": 0,
    "tabs": []
}
```

Profiles

The Profiles endpoint allows you to retrieve information about the available Profiles, Channels and Engines.

GET /api/v1/profiles

Returns the entire list of profiles available in Viz Arc.

curl -s -X GET http://localhost:9004/api/v1/profiles | python -m json.tool

```
[
    {
        "name": "Local",
        "channels": [
            {
                "name": "Local",
                "type": 1,
                "engines": [
                    {
                        "name": "localhost"
                    }
                ],
                "model": {
                    "engineConfigTuple": [
                        {
                            "item1": {
                                "name": "localhost",
                                "ipAddress": "127.0.0.1",
                                "port": 6100,
                                "launcherPort": 5644,
                                "webInterfacePort": 30010,
                                "webSocketPort": 30020,
                                ...
```

GET api/v1/profiles/{id | name}

Returns the profile specified by the *id* or *name*.

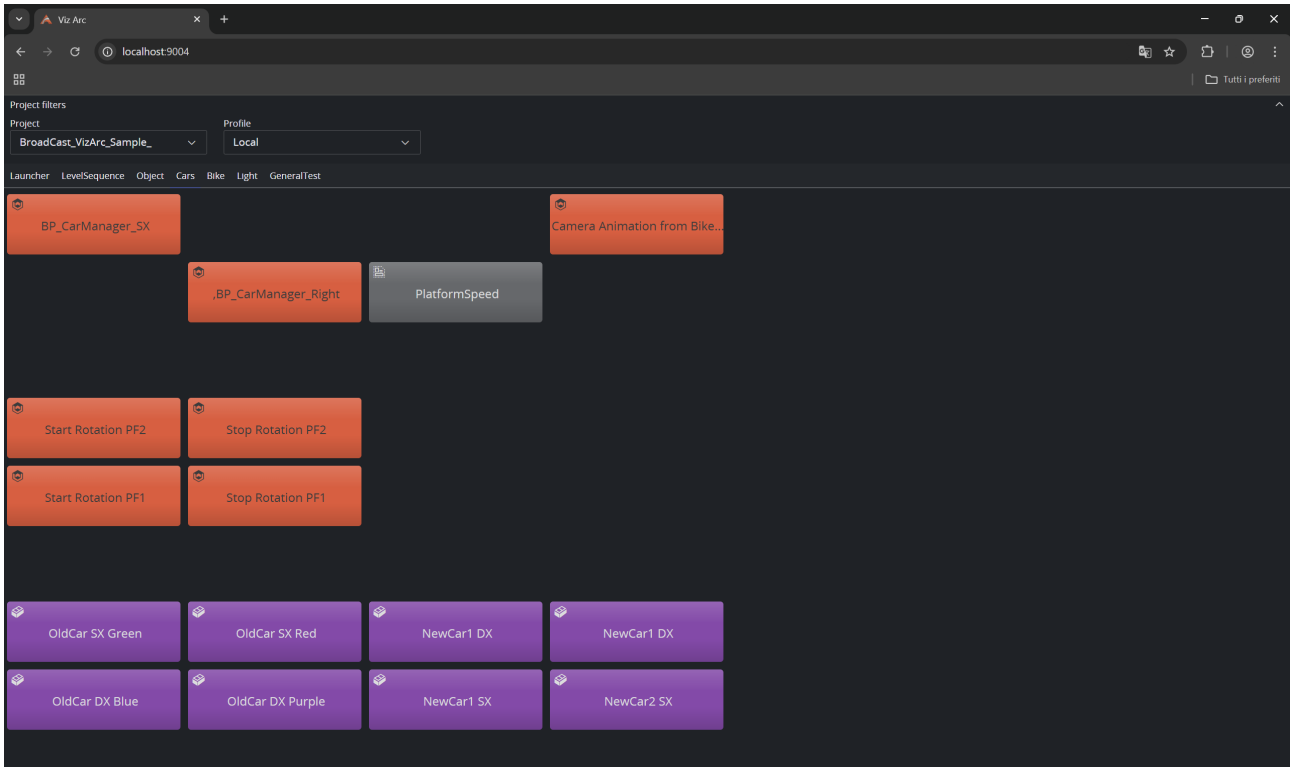
curl -s -X GET http://localhost:9004/api/v1/profiles/0 | python -m json.tool

```
{
    "name": "Local",
    "channels": [
```

```
{
    "name": "Local",
    "type": 1,
    "engines": [
        {
            "name": "localhost"
        }
    ],
    "model": {
        "engineConfigTuple": [
            {
                "item1": {
                    "name": "localhost",
                    "ipAddress": "127.0.0.1",
                    "port": 6100,
                    "launcherPort": 5644,
                    "webInterfacePort": 30010,
                    "webSocketPort": 30020,
                    ...
```

Web Application Sample Usage

Once the Viz Arc REST service is installed, it also hosts a Web Application that uses some of the APIs described above.



By selecting a **Project** and a **Profile**, we can see the Project's structure. By clicking on an action, Viz Arc REST Service executes it.

Copyright © 2025 Vizrt

Keyer

The **Keyer Enpoint** is intended to be used to control the currently loaded **precision keyer** on a given output channel, through the external Loupedeck device. It allows to select a channel, restore a temporary preset and to restore the original action. It is intended to be used together with the **Executer** endpoint.

select

POST api/v1/Keyer/select

Selects the active keyer on a given channel (*profileId* and the *channel* must be specified on the POST call). When the parameter *execute* is true (its default value), the action is executed as well, thus sending the keyer values on the engine.

curl -s -H 'Content-Type: application/json' -d "{ 'profileId': 'Profile-1', 'channel' : 'cam1', 'execute' : false }" -X POST http://localhost:9004/api/v1/Keyer/select

When a Loupedeck device is connected, it shows the current keyer values of the selected channel.

```
{
  "profileId" : "Profile-1",
  "channel" : "cam1",
  "execute" : false
}
```

Select the last loaded keyer that was executed on *Profile-1,* on channel *cam1*. If case no action has been previously loaded through the *Executer* endpoint, this call returns with BadRequest.

save

The save endpoint allows to save a temporary keyer setting on the REST server. Please note that the temporary settings are going to be lost after a REST server restart. The temporary settings are associated to the profile and channel, so each combination of profile and channel can potentially have one temporary “preset”.

curl -s -H 'Content-Type: application/json' -d "{ 'profileId': 'Profile-1', 'channel' : 'cam1' }" -X POST http://localhost:9004/api/v1/Keyer/save

```
{
  "profileId" : "Profile-1",
  "channel" : "cam1",
}
```

Makes a copy of the current keyer loaded on profile *Profile-1* and channel *cam1*. It is stored as a temporary preset and can be recalled through the **restoreTemp** preset.

Copyright © 2025 Vizrt

restoreTemp

Restores the previously stored keyer preset, loads it on the respective output engine and loads it on the external control device (for example, Loupedeck).

curl -s -H 'Content-Type: application/json' -d "{ 'profileId': 'Profile-1', 'channel' : 'cam1' }" -X POST http://localhost:9004/api/v1/Keyer/restorePreset

```
{
  "profileId" : "Profile-1",
  "channel" : "cam1",
}
```

Restores the temporarily saved keyer preset loaded on profile *Profile-1* and channel *cam1*.

restoreOriginal

Restores the original keyer preset, loads it on the respective output engine, and loads it on the external control device (for example, Loupedeck). The original preset is the action that was executed via the **Executer** endpoint.

curl -s -H 'Content-Type: application/json' -d "{ 'profileId': 'Profile-1', 'channel' : 'cam1' }" -X POST http://localhost:9004/api/v1/Keyer/restorePreset

```
{
  "profileId" : "Profile-1",
  "channel" : "cam1",
}
```

Restores the original keyer preset loaded on profile *Profile-1* and channel *cam1*.

status

Returns the status of the currently loaded keyer. It returns whether the keyer has been modified compared to the original keyer that was executed through the **Executer** endpoint.

curl -s -H 'Content-Type: application/json' -d "{ 'profileId': 'Profile-1', 'channel' : 'cam1' }" -X POST http://localhost:9004/api/v1/Keyer/status

```
{
  "profileId" : "Profile-1",
  "channel" : "cam1",
}
```

Checks the status of the keyer preset loaded on profile *Profile-1* and channel *cam1*. The response to the request looks as follows:

Copyright © 2025 Vizrt

```
 {
    success = true,
    reason = "",
    modified = true,
}
```

- *success* is true when a comparison was able to be processed.
- *reason* is empty on success, otherwise it contains an error message.
- *modified* is true when the current keyer settings are different from the original keyer settings.

5.7 Data Integration

- Singal-R
  - C# Client Sample
  - Viz Arc Script
- MQTT Broker
  - Sample C# Snippet for Publishing a MQTT Topic to Viz Arc
  - Viz Arc JavaScript Sample to Receive and Read MQTT Topics through the DataMap
- TCP Server
  - Set DataMap
  - Get DataMap
  - Execute Action
- UDP Server
- DataMap Linking
  - JSON Sample Data
  - Dropdowns
  - Tables
- DataMap Websocket
- Excel and CSV integration

5.7.1 Singal-R

When running Viz Arc, a SignalR Hub waits for incoming connections internally. The Viz Arc **In-App Web API** portcan be configured, see Communication under the General Settings. The hub where the various methods can be invoked by clients is called **VizArcHub**.

> **Information:** To use an external application running a SignalR client, please use a .NET 8.0 based project.

The server supports the following methods

- **setStringVariable** (string varName, string value)
- **setJSONVariable** (string varName, string json)

Those two methods allow clients to set simple variable names with either a string value or a JSON string. Viz Arc scripts is notified of variable changes and can process their value. The following code sample shows client setting variables in Viz Arc:

C# Client Sample

```
using Microsoft.AspNetCore.SignalR.Client;

// connects to VizArcHub on localhost and default port 5004
 HubConnection connection = new HubConnectionBuilder()
   .WithUrl("http://127.0.0.1:5004/signalr/VizArcHub")
   .Build();
```

Copyright © 2025 Vizrt

```
// connect
await connection.StartAsync();

// set some DataMap variable
await connection.InvokeAsync("setStringVariable", "hello", "world!");
```

In any Viz Arc script, the values above can be read, as in the following simple sample:

Viz Arc Script

```
Global.OnInit = function () {
  Console.WriteLine( "data " + GetData("hello"))
  Console.WriteLine( "json " + GetData("jsonData").ToString())
  Console.WriteLine( "name is " + GetData("jsonData").Name.ToString())

  // subscribe to all DataMap variable changes
  SubscribeDataMap("")
}

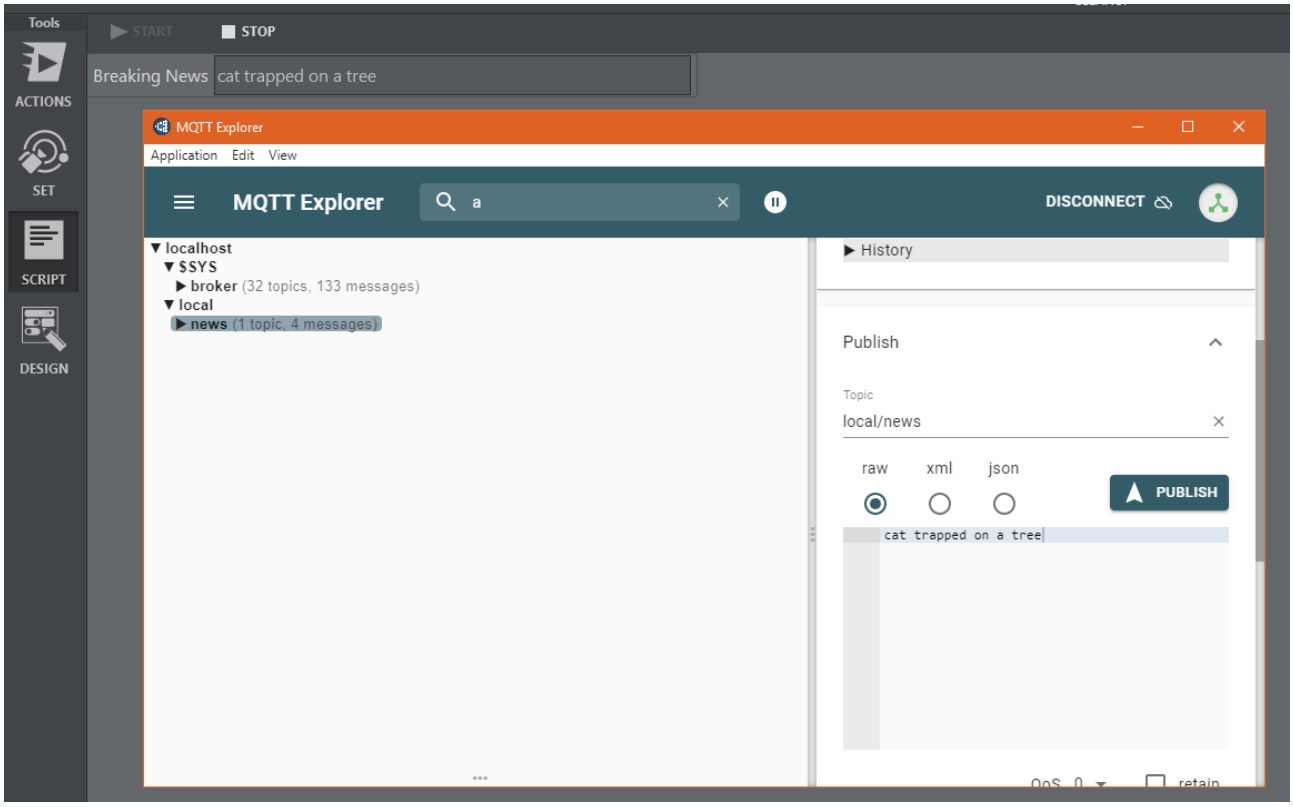
Global.OnDataMapValueChanged = function (varName) {
  Console.WriteLine( varName + " changed to "+ GetData(varName).ToString())
  if( varName == "jsonData" )
    Console.WriteLine( "nams is " + GetData("jsonData").Name.ToString())
}
```

The method **GetData** returns the currently set value of the variable.

5.7.2 MQTT Broker

When the MQTT broker port is configured, an internal MQTT broker starts on Viz Arc startup. All topic changes are transmitted to the DataMap using the MQTT topic as the key and the topic's payload as value.

Copyright © 2025 Vizrt



To test the integration you can wither use the **MQTT Explorer** (this tool can be useful to monitor Viz Arc's MQTT broker as well) or a .NET core project using the following C# code:

Sample C# Snippet for Publishing a MQTT Topic to Viz Arc

```
// using nuget package MQTTnet.AspNetCore and MQTTnet.Extensions.ManagedClient
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Client.Options;

// ...

var factory = new MqttFactory();
IMqttClient mqttClient = factory.CreateMqttClient(); // create MQTT client

var options = new MqttClientOptionsBuilder().WithTcpServer("localhost",
1883).Build(); // create connection options
var connResult = mqttClient.ConnectAsync(options, CancellationToken.None).Result; //
connect to VizArc's MQTT broker

var message = new
 MqttApplicationMessageBuilder().WithAtLeastOnceQoS().WithTopic("local/news/
breaking").WithPayload("breaking news").Build(); // create message
```

Copyright © 2025 Vizrt

```
var result = mqttClient.PublishAsync(message, CancellationToken.None).Result; //
publish the message
```

Viz Arc JavaScript Sample to Receive and Read MQTT Topics through the DataMap

```
var listenTopic = "local/news/breaking"

Global.OnInit = function ()
{
    SubscribeDataMap(listenTopic)
}

Global.OnDataMapValueChanged = function (varName)
{
    if( varName == listenTopic )
        news.Value = GetData(listenTopic)
}
```

The script above listens to changes on the "local/news/breaking" topic and updates a text box accordingly.

5.7.3 TCP Server

Viz Arc starts up a TCP server which can be used to set and get DataMap variables. The port of the server can be configured, see General Settings.

You can set DataMap variables using a command string from a Viz Engine script:

Set DataMap

```
System.TcpSendAsync("resultSetDM", "localhost", 9204, "ARC*DATAHUB SET key|someValue
1.0 1.0 1.0", 1000)
```

The command above sets the variable *key* to the value *someValue 1.0 1.0 1.0*. The **|** character separates the key from the value. This value can then be further processed in Viz Arc.

To read back a value from the DataMap in Viz Arc you can use the *GET* command like in the sample below.

Get DataMap

```
Dim ret = System.TcpSend("localhost", 9204, "ARC*DATAHUB GET key", 1000)
```

The Viz Engine script variable *ret* contains the value of *key*.

You can execute an action using a command string from a Viz Engine script:

Copyright © 2025 Vizrt

Execute Action

```
System.TcpSendAsync("resultSetActionName", "localhost", 9204, "ARC*ACTION EXECUTE
loadSceneA", 1000)
System.TcpSendAsync("resultSetActionGUID", "localhost", 9204, "ARC*ACTION EXECUTE
49c525fa-0ce8-46e1-bae4-acfb4abdb629", 1000)
```

The commands above search for actions **loadSceneA** by name and **49c525fa-0ce8-46e1-bae4-acfb4abdb629** by GUID and execute them if they can be found in the current project.

5.7.4 UDP Server

Viz Arc starts up an internal UDP server used to set DataMap variables. The port of the server can be configured, see General Settings (default `9304`).

The syntax to be used when updating or setting a DataMap variable is simple:

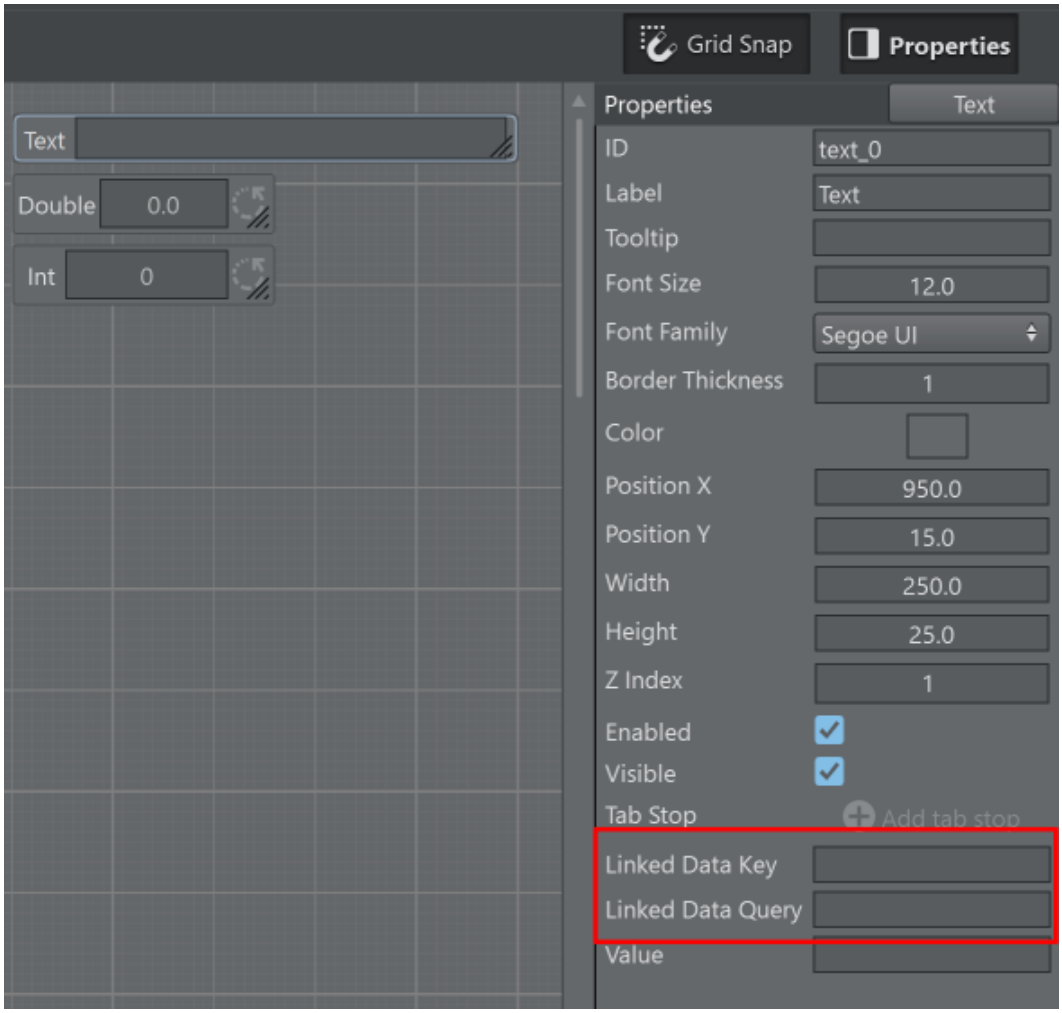
key|value `\0`

The pipe symbol | is used to separate the key from the value. A message is terminated with the character \0.

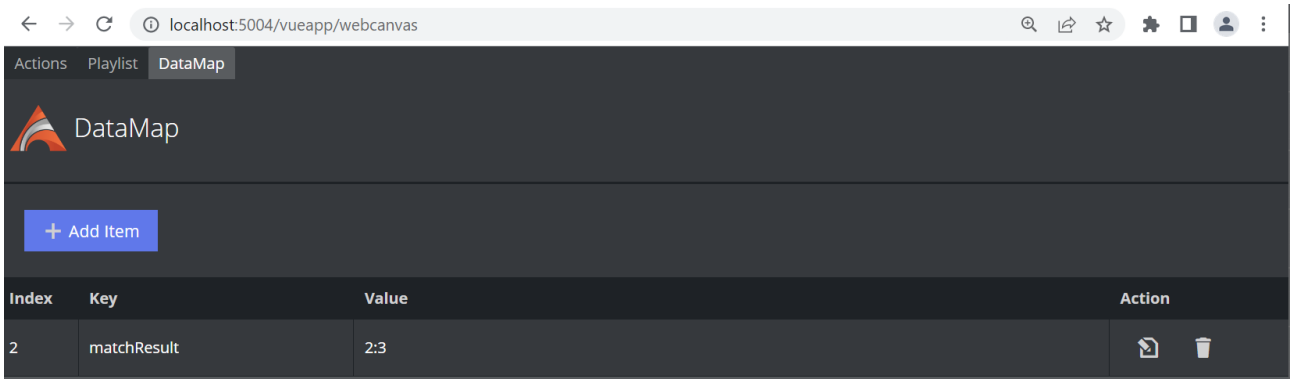
5.7.5 DataMap Linking

Some scripting UI components like **Text**, **Int** or **Double** have the properties **Linked Data Key** and **Linked Data Query**.

Copyright © 2025 Vizrt

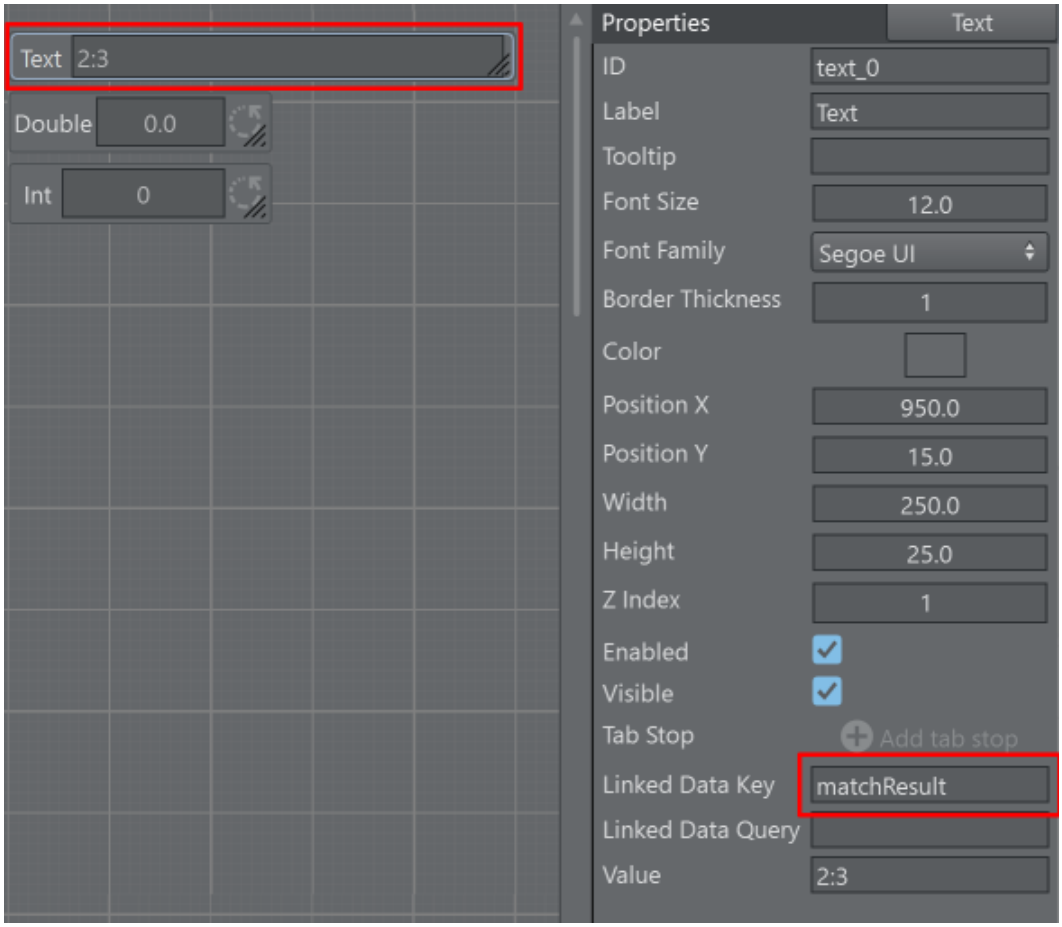


The **Data Key** entry can accept any key name used in a **DataMap** entry. As an example add a key *matchResult* using the web UI to insert key/value pairs on Viz Arc.



Enter *matchResult* as **Linekd Data Key** and the text value updates to the previously entered data.

Copyright © 2025 Vizrt



For more complex data structures such as *.json* strings, the **Linked Data Query** can be used.

JSON Sample Data

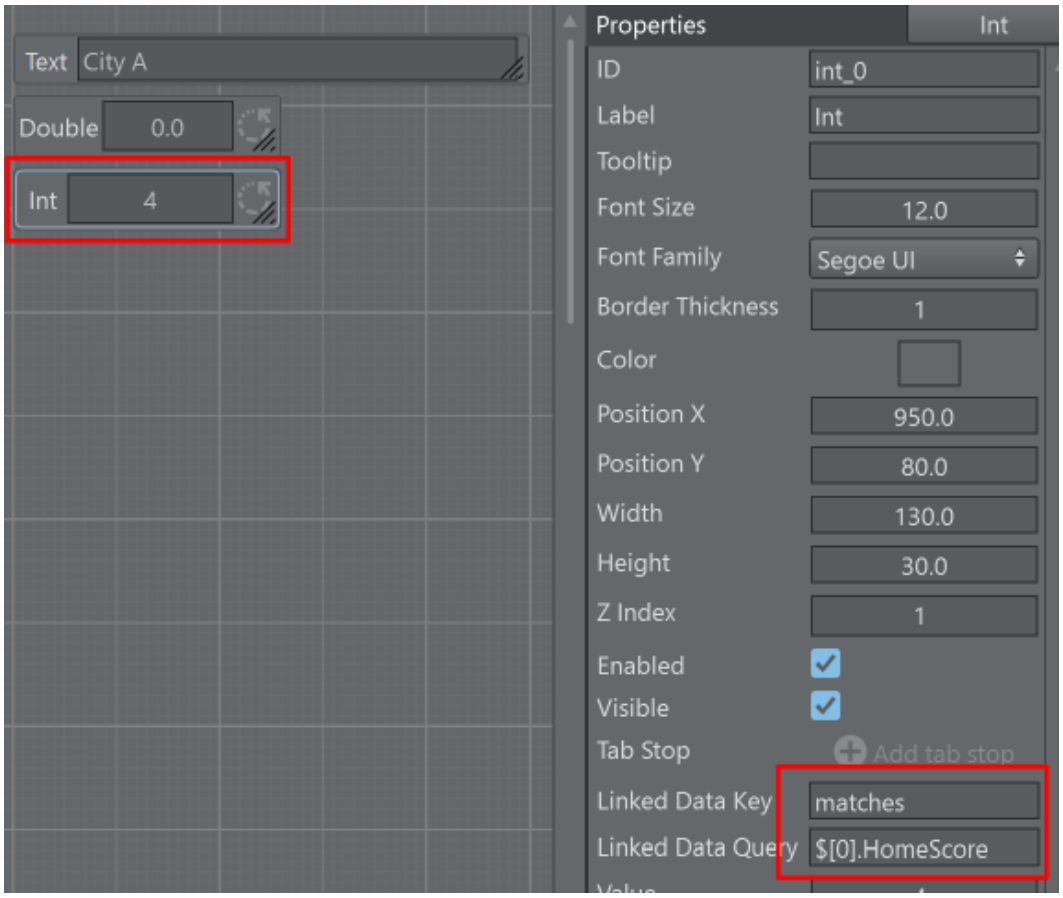
```
[
   {
      "matchID":4242,
      "Home":"City A",
      "Away":"City B",
      "HomeScore":4,
      "AwayScore":2
   },
   {
      "matchID":4243,
      "Home":"FC A",
      "Away":"FC B",
      "HomeScore":0,
      "AwayScore":1
   }
]
```

Copyright © 2025 Vizrt

The DataMap after adding the above *.json* using the **matches** key.

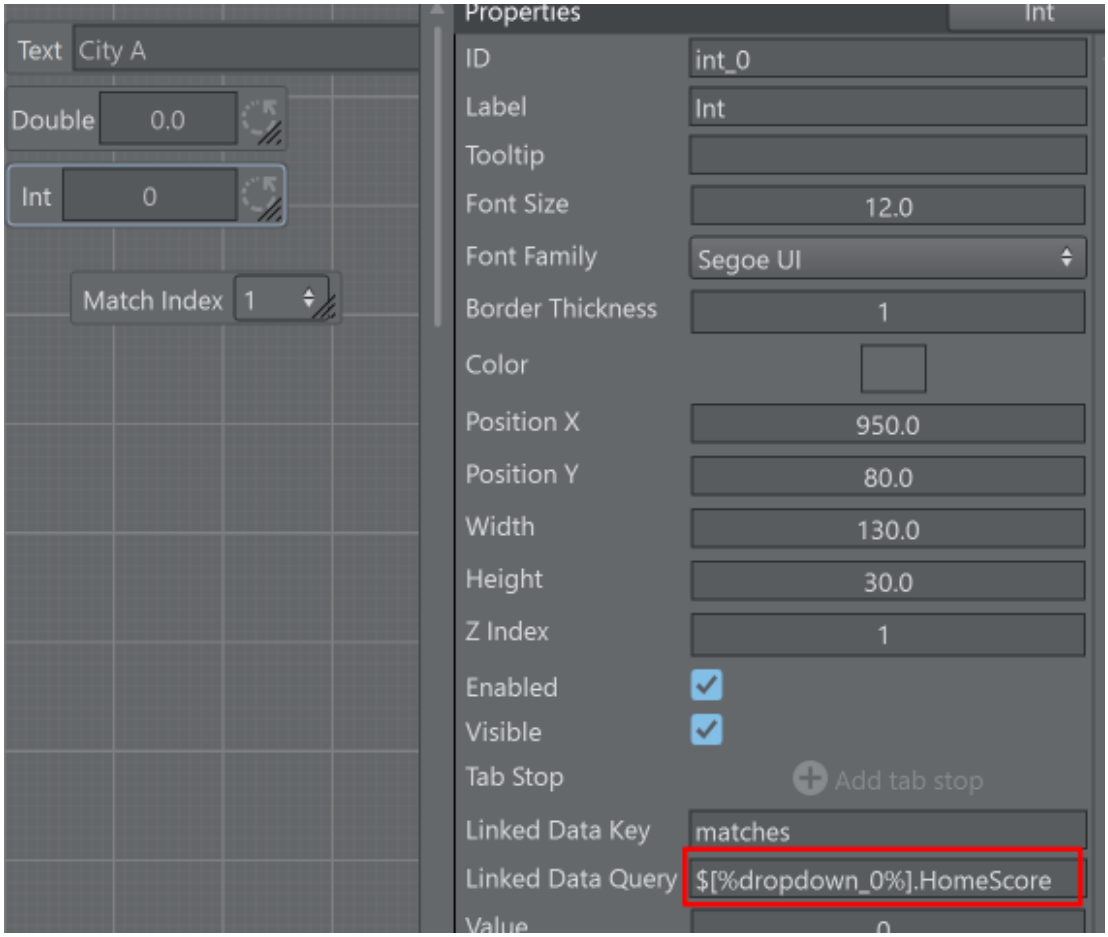


Enter a **JSONPath** string *$[0].HomeScore* to get the home score of the first *.json* array element.



Instead of using constant values for indexing the *.json* array a variable can be used instead. A variable must be enclosed in **%** character and should be the ID of an existing scripting UI element.

Copyright © 2025 Vizrt



$[%dropdown_0%].HomeScore gets resolved to *$[1].HomeScore* as the dropdown value of the **dropdonw_0** component is 1.

In order to force an evaluation of the expression of the **Linked Data Query**, use the function **UpdatateDataLink** available on all scripting parameters.

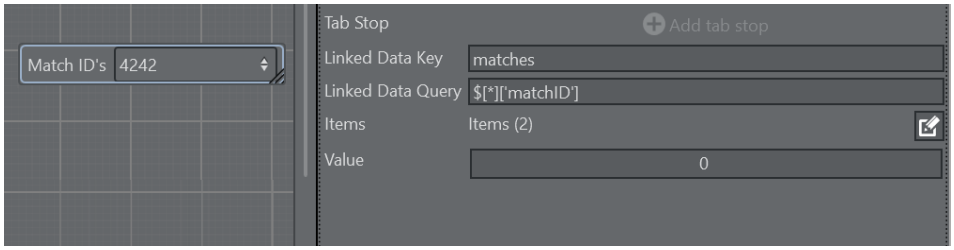
The following sample forces a re-evaluation of the Linked Data Query on parameter *int_0* whenever the value of the dropdown *dropdown_0* changes.

```
Global.OnParameterChanged = function (id)
{
    if( id == "dropdown_0" )
        int_0.UpdateDataLink()
}
```

Dropdowns

Dropdowns can be populated using Queries that return arrays, for example:

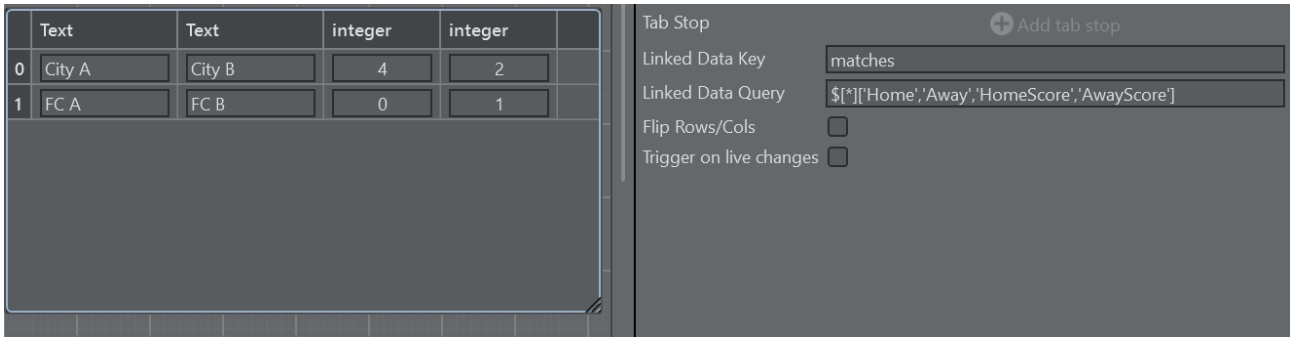
Copyright © 2025 Vizrt



In this case the query **$[*]['matchID']** selects the matchID's of all array elements.

Tables

Tables can be populated in a similar manner as dropdowns:



The query **$[*]['Home','Away','HomeScore','AwayScore']** returns an array containing the four selected members Home, Away, HomeScore and AwayScore in this order. The calculated array looks like:

```
["City A", "City B", 4, 2, "FC A", "FC B", 0, 1]
```

It is important to note that the Table needs to have the matching columns in place, in this sample the table has been prepared to have the first two columns to be strings and the last two columns to be integer types.

The table is then populated row by row trying to match the resulting array of the query.

To learn more about the **JSONPath** syntax, please refer to https://goessner.net/articles/JsonPath/.

5.7.6 DataMap Websocket

A client can connect to Viz Arc's DataHub socket on the endpoint ws://hostname:5004/api/v1/ws, where hostname is the host name where Viz Arc is running. You can subscribe to DataMap variable changes by sending a JSON string like in the following examples.

To register to changes of **someDataKey** send a JSON setting like this through the websocket:

```
{
    "type" : "DataMapSubscribe",
    "value" : "someDataKey"
}
```

To register to all changes of **all** keys use an empty string as value.

Copyright © 2025 Vizrt

```
{
    "type" :"DataMapSubscribe",
    "value" :""
}
```

Whenever a DataMap variable changes you receive a JSON answer on the websocket like this:

```
{"type":"DataMap","value":{"key":"someDataKey","value":"Hello World some data"}}
```

5.7.7 Excel and CSV integration

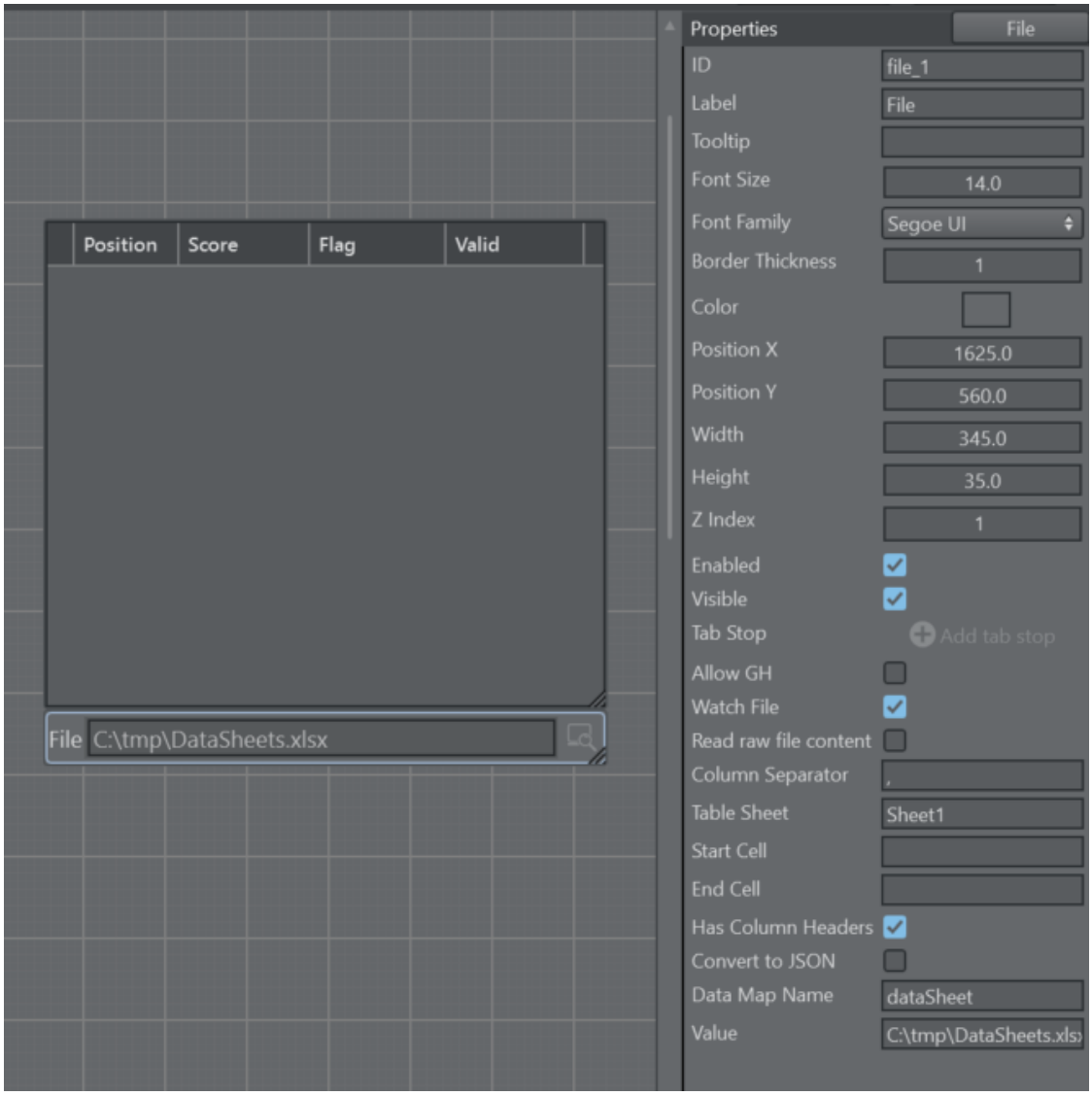
The **File Parameter** script element has a couple of properties which allow you to watch an Excel or *.csv* file and load it whenever there are changes to the contents of the file. It can load the raw contents of the file or the containing table can be converted into a JSON format. The data is stored in both cases into the **DataMap**.

Starting from a Excel Sheet, define for example column headers **Valid**, **Position** and **Score** and enter some rows with values.

	A	B	C	D
1	Valid	Position	Score	
2	0	1	1.1	
3	1	2	2.2	
4	FALSE	3	3.3	
5	TRUE	4	4.4	
6				
7				

Add in a template or the global script a **File Parameter** and a **Table** UI element. Select an Excel or *.csv* file and check the flag **Has Column Header** to indicate that the first row contains the headers. Check the **Watch File** option to update the data anytime the file is saved to disk. Specify a variable key of your choice in **Data Map Name**.

Copyright © 2025 Vizrt



- **Read raw file content**: Reads the plain file content without processing it.
- **Column Separator**: Defines the column separator for *.csv* files. Autodetects the separator when blank.
- **Table Sheet**: Specifies the Excel sheet to read. You can use the string name of the table or the index of the sheet (starting from 1). If the field is left empty, the first sheet is read.
- **Start Cell**: Specifies a row/column offset using Excel style syntax where the table content starts (for example, **C2** would start to read from row 2 and column 3). If left empty, row/column is 1.
- **End Cell**: Specifies a row/column offset using Excel style syntax where the table content ends (for example, limit the table to cell **F9** would mean to crop the data at column 6 and row 9). Auto-detects the row and column count when empty.
- **Has Column Header**: Specifies whether the the table contains header information in the first row. Uncheck if the table does not contain header information.
- **Convert to JSON**: Converts the table into a JSON format. The above example would be converted into this:

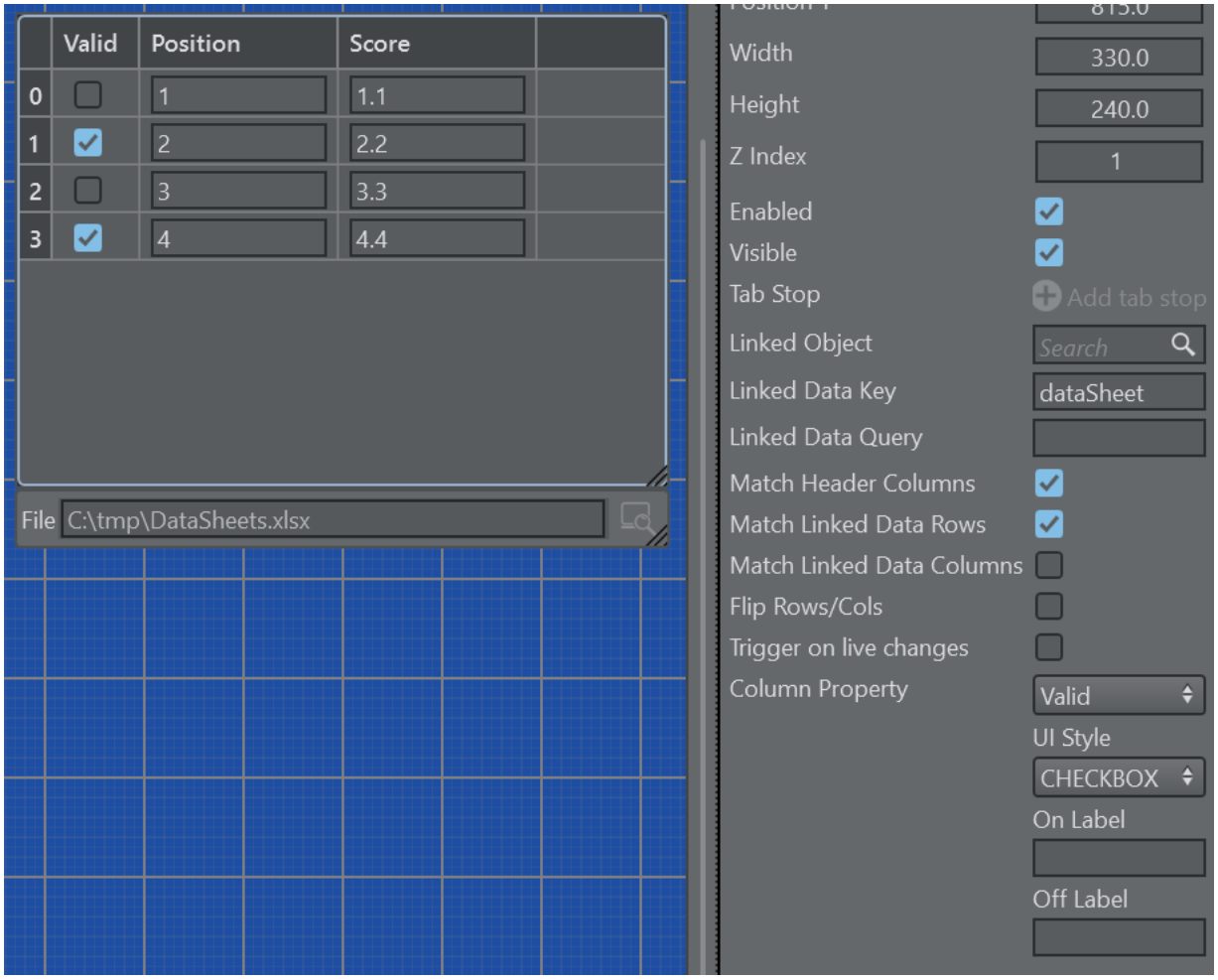
```
[
  {
```

Copyright © 2025 Vizrt

```
    "Valid": 0,
    "Position": 1,
    "Score": 1.1
  },
  {
    "Valid": 1,
    "Position": 2,
    "Score": 2.2
  },
  {
    "Valid": false,
    "Position": 3,
    "Score": 3.3
  },
  {
    "Valid": true,
    "Position": 4,
    "Score": 4.4
  }
]
```

Create a **Table** parameter and check the flag **Match Linked Header Columns** in order to match the column header names of the file with the column names of the scripting table. Otherwise, the columns are matched in order. Check the option **Match Linked Data Rows** to match the number of rows of the data sheet with the number of rows in the script table. Check the **Match Linked Data Columns** flag to create automatically the columns as they are in the excel sheet. This will not work when **Match Header Columns** is enabled. Specify the same **DataMap** key that has been previously specified in the File Parameter in **Linked Data Key**.

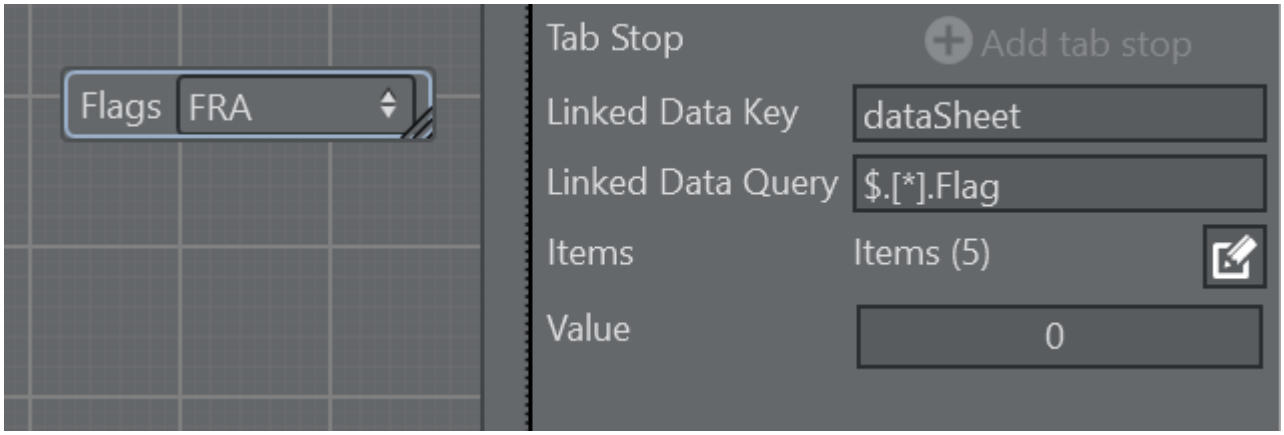
Copyright © 2025 Vizrt

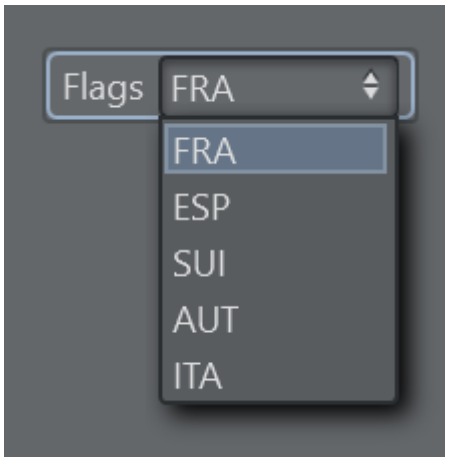


When you choose to convert the table into a **JSON**, you can use the **Linked Data Query** e.g. for a **DropDown** component to populate those elements with a column of a table:

	A	B	C	D
1	Valid	Flag	Position	Score
2	TRUE	FRA	1	1.1
3	FALSE	ESP	2	2.2
4	FALSE	SUI	3	3.3
5	TRUE	AUT	4	4.4
6	FALSE	ITA	5	5.5

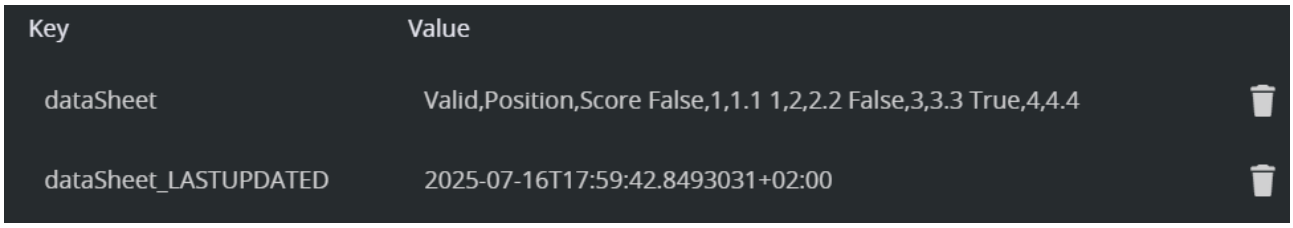
Copyright © 2025 Vizrt





Using a query like `$.[*].Flag` as **Linked Data Query** populates the **DropDown** with a list of Flags specified in the Excel sheet.

On the **DataMap** an additional key is generated with a postfix **_LASTUPDATED** containing a Date and Time string indicating the time when the data has been last read.



The date string can be easily converted to a Date object within a Javascript script:

```
let dt = new Date(GetData("dataSheet_LASTUPDATED")) // create a date object
Console.WriteLine("updated sheet at: " + dt.getHours() + ":" + dt.getMinutes() + ":"
 + dt.getSeconds() )
```

Copyright © 2025 Vizrt

5.8 Integrations

- Art-Net DMX Integration
- Loupedeck
- MOS Integration
- Stream Deck
- Viz Arena
- Working with Unreal Engine
- Timecode from Plura or NDI

Copyright © 2025 Vizrt

5.8.1 Art-Net DMX Integration

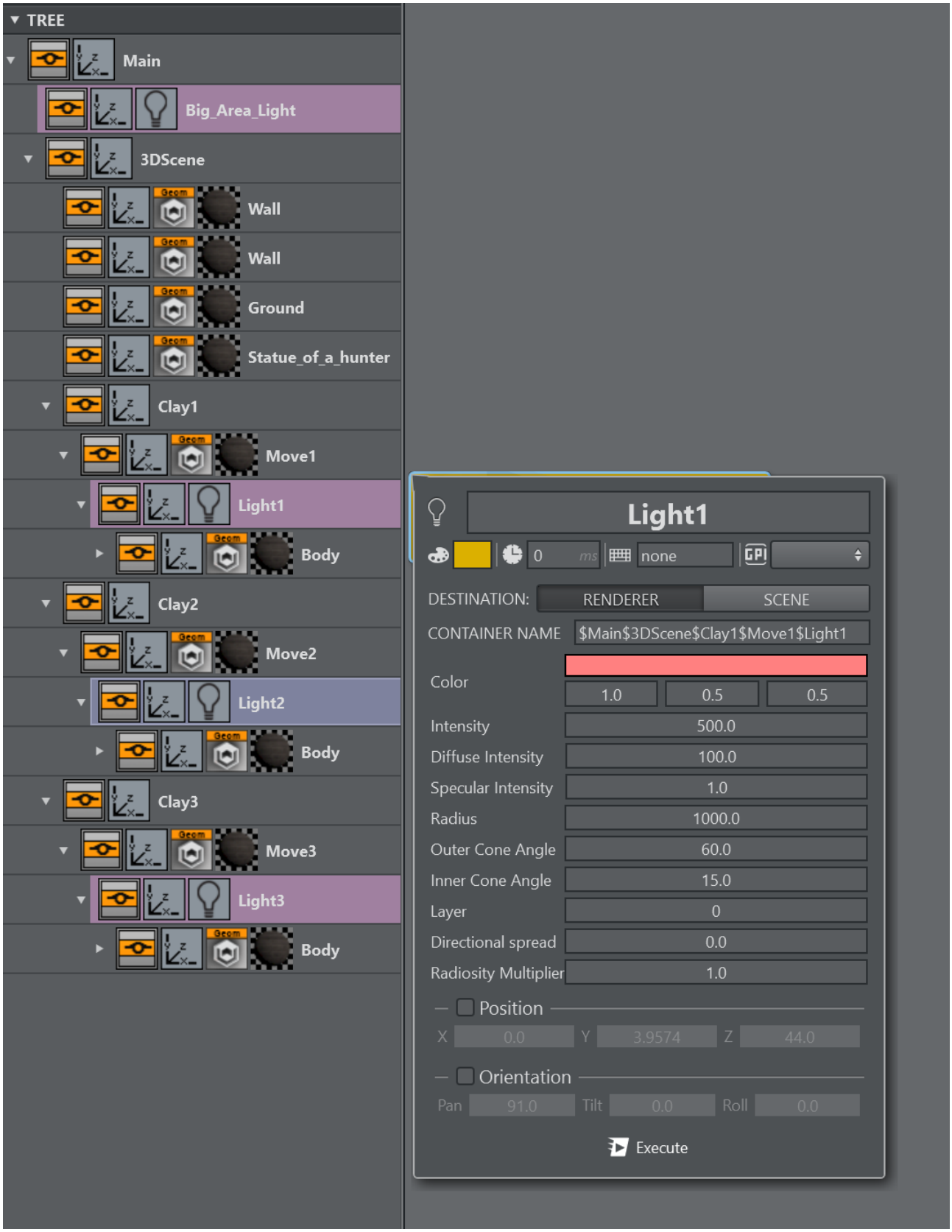
Art-Net is a data distribution protocol that allows DMX lighting data to be transported over an ethernet network.

- Usage with Light Action
- Usage with Precision Key Action
- Usage with Template Actions
- Enabling and Disabling DMX Events
- Scripting

Usage with Light Action

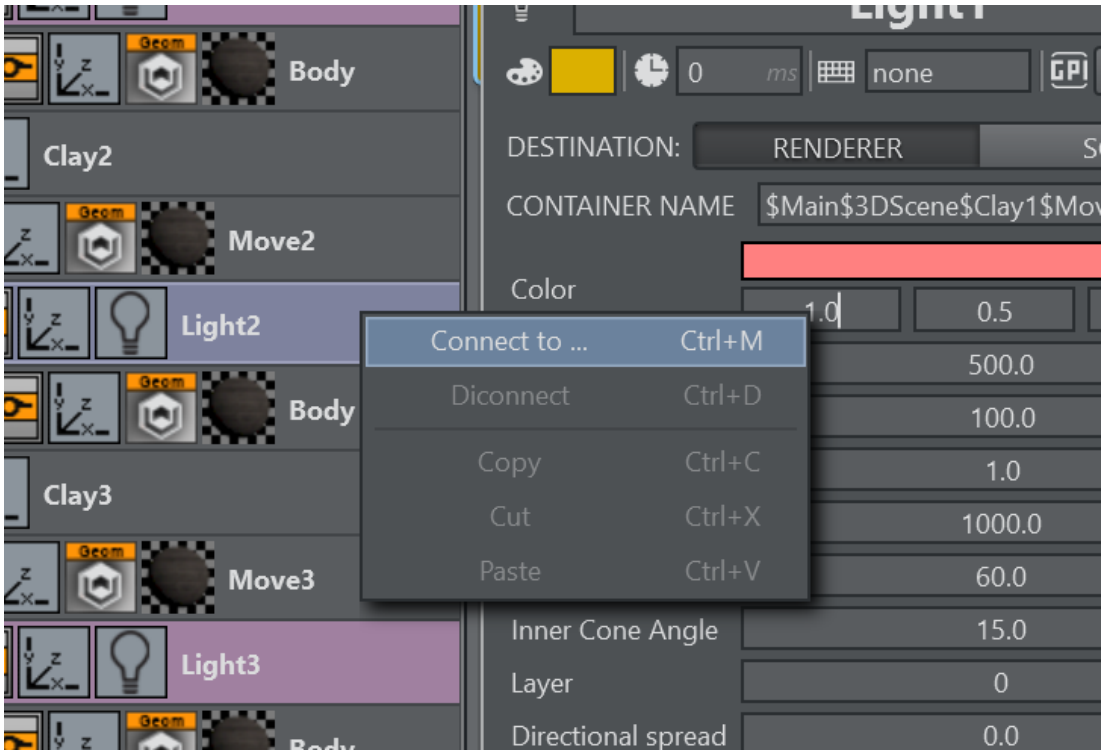
Select a scene containing Viz Engine version 5 based lights and load it into Viz Arc. Drag the light from the scene tree to the action canvas to create a Light Action.

Copyright © 2025 Vizrt

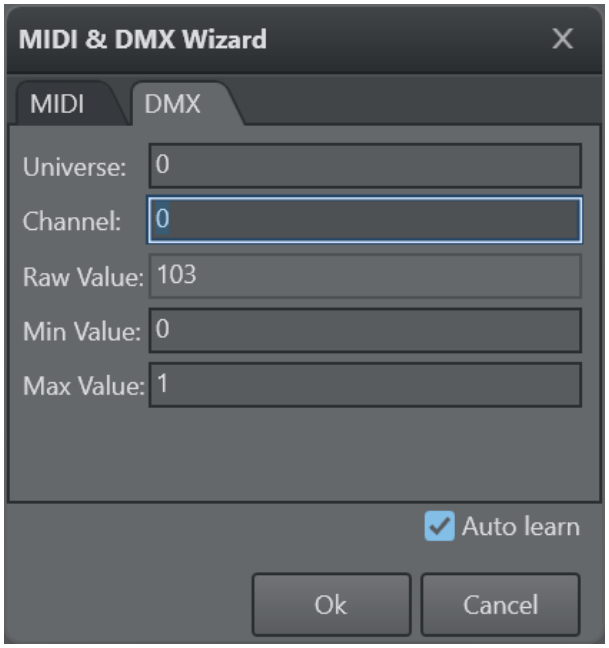


Copyright © 2025 Vizrt

Open the context menu and select the **Connect to ...** entry.



In the wizard window that opens select the **DMX** tab.



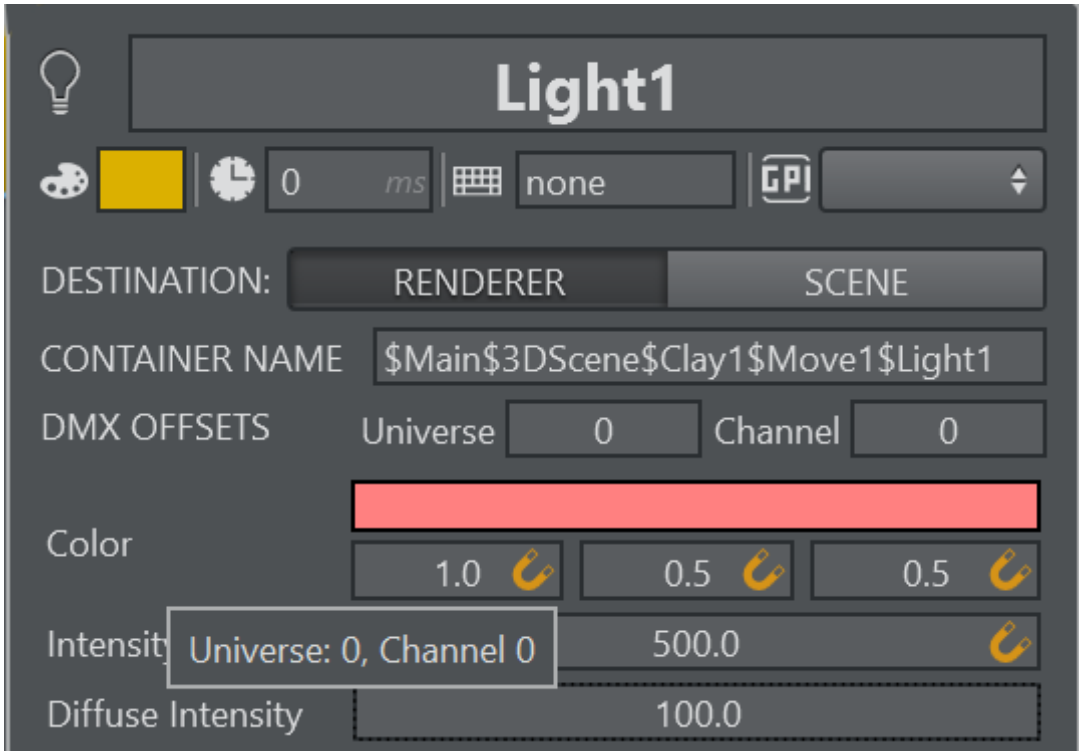
When the **Auto learn** option is enabled, the panel displays the last **Universe** and **Channel** that has changed. It can be used to easily identify a hardware slider or a knob.

The **Raw Value** displays the received raw value in the range of [0..255]). The **Universe** and **Channel** values can be entered manually as well.

Copyright © 2025 Vizrt

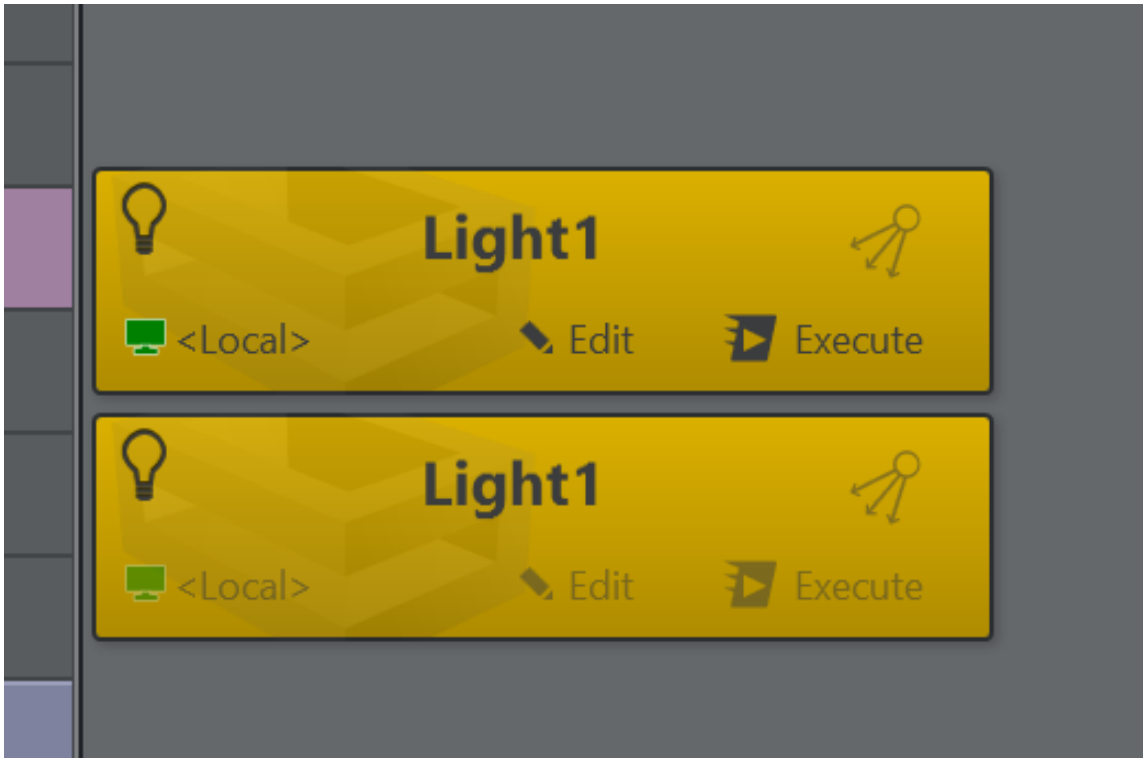
The **Min/Max Value** determine the range where this channel is to be mapped.

Once a property has been mapped, a small horseshoe magnet icon indicates the presence of a mapping. A tooltip shows the mapped Universe and Channel.

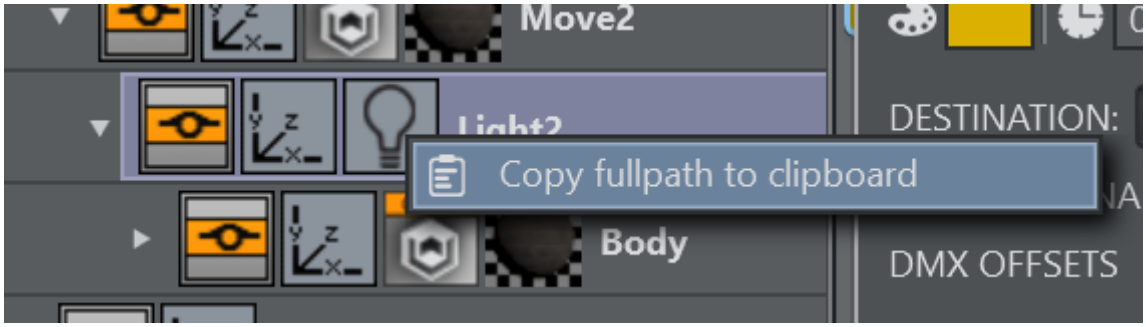


In case more light need to be mapped in a similar fashion as the first one, the easiest approach is to duplicate the action and configure DMX offsets on the duplicated action.

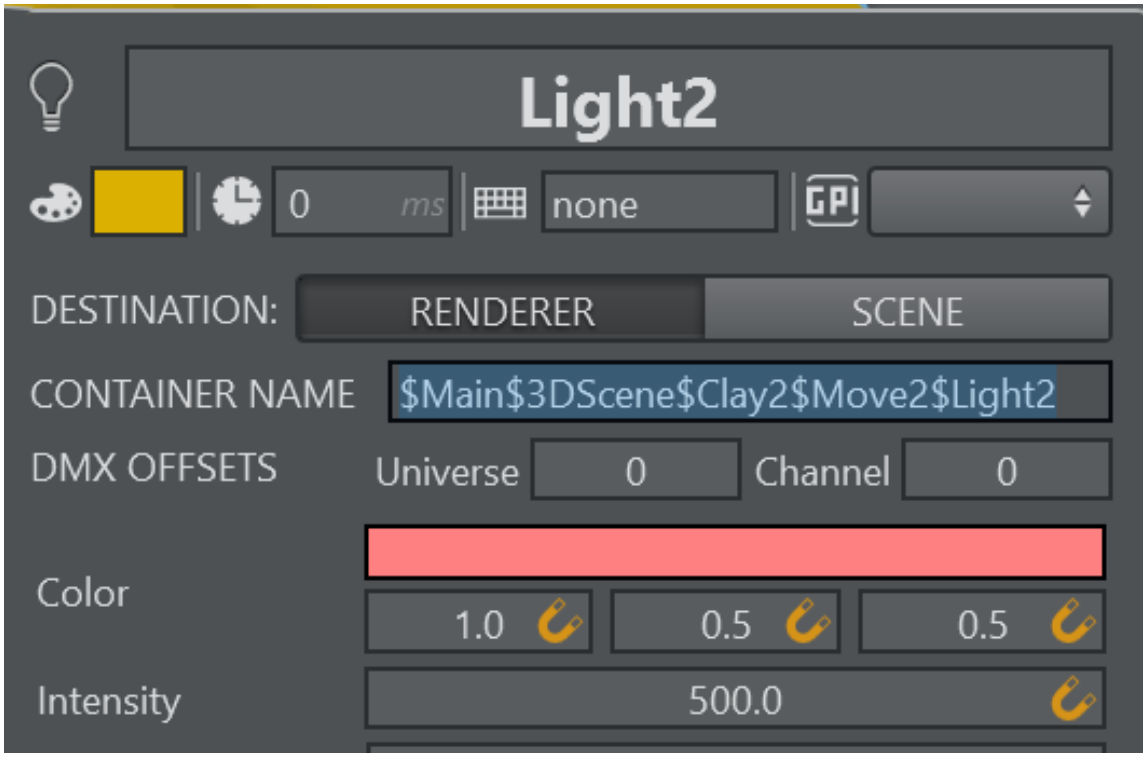
Copyright © 2025 Vizrt



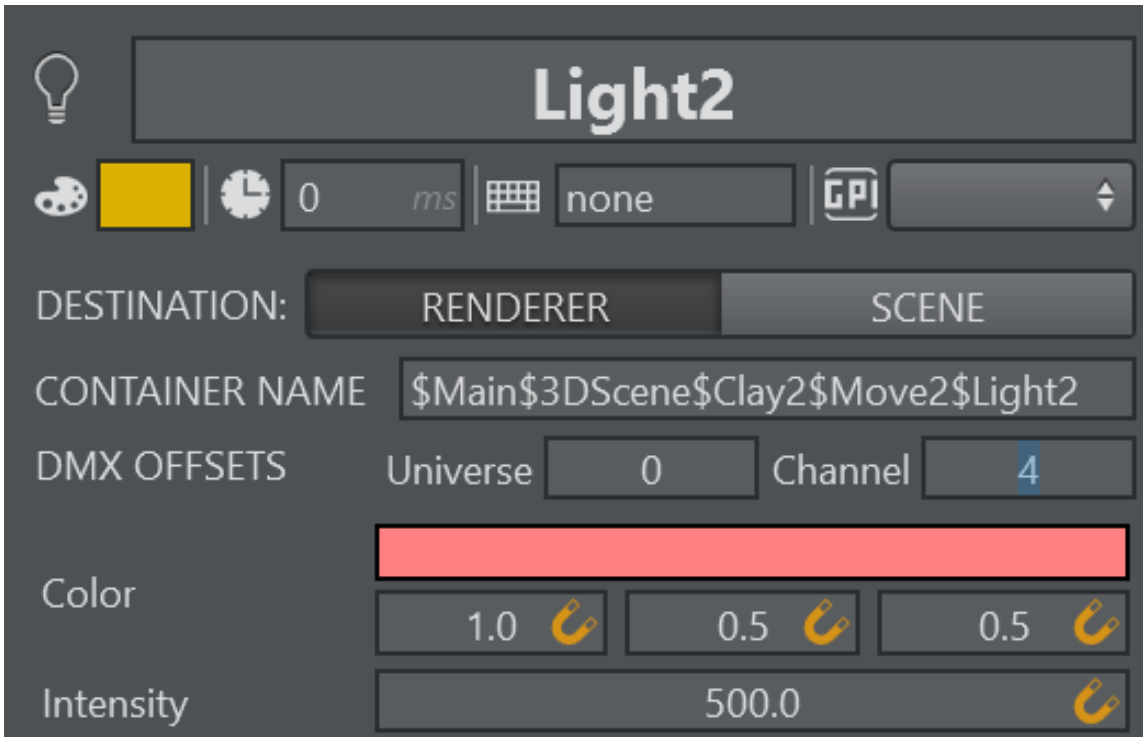
Copy the full path of the second light either by copying it from the scene tree directly or enter the tree path manually.



Copyright © 2025 Vizrt

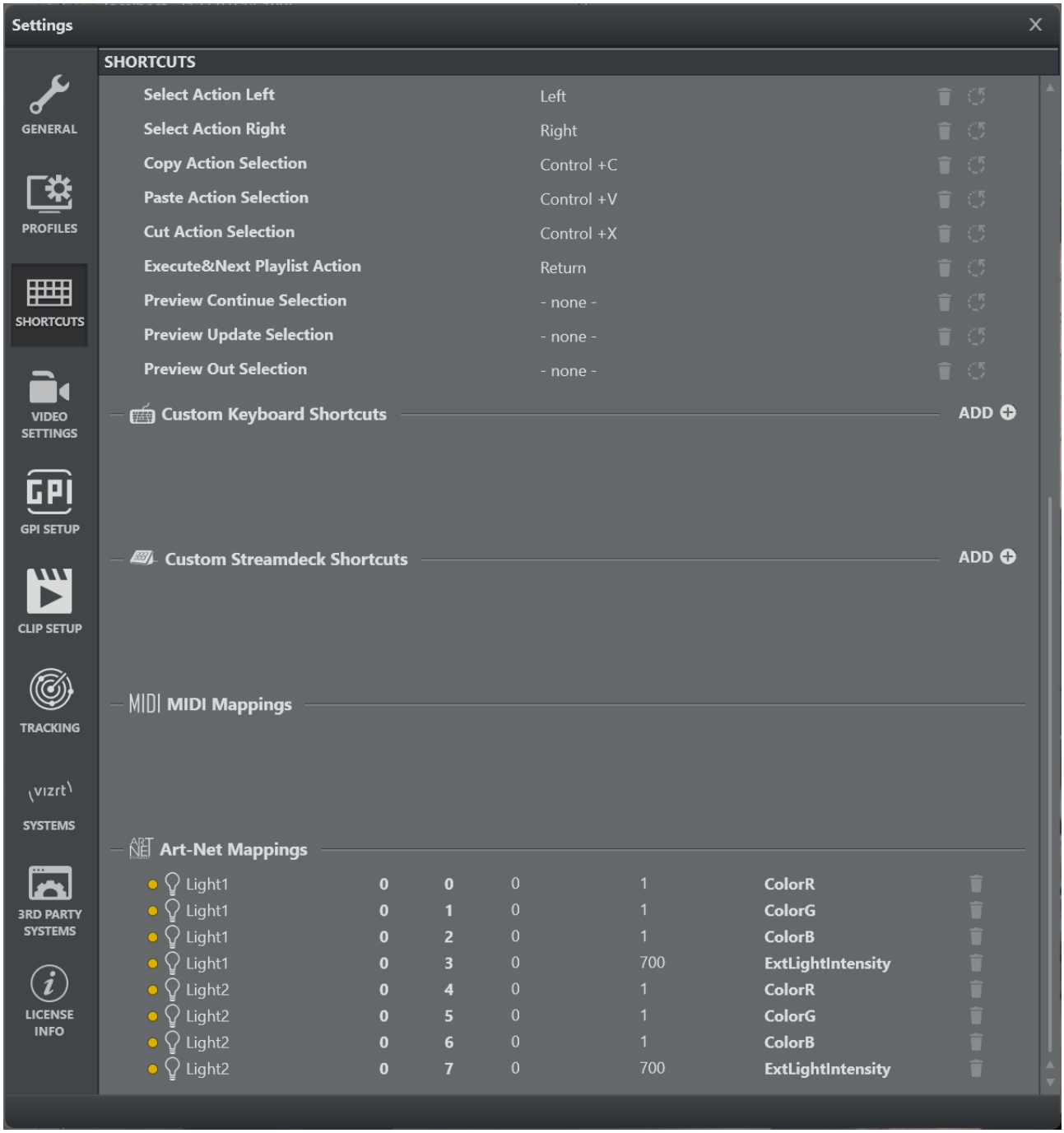


Enter an appropriate Universe/Channel offset the **DMX OFFSET** line.



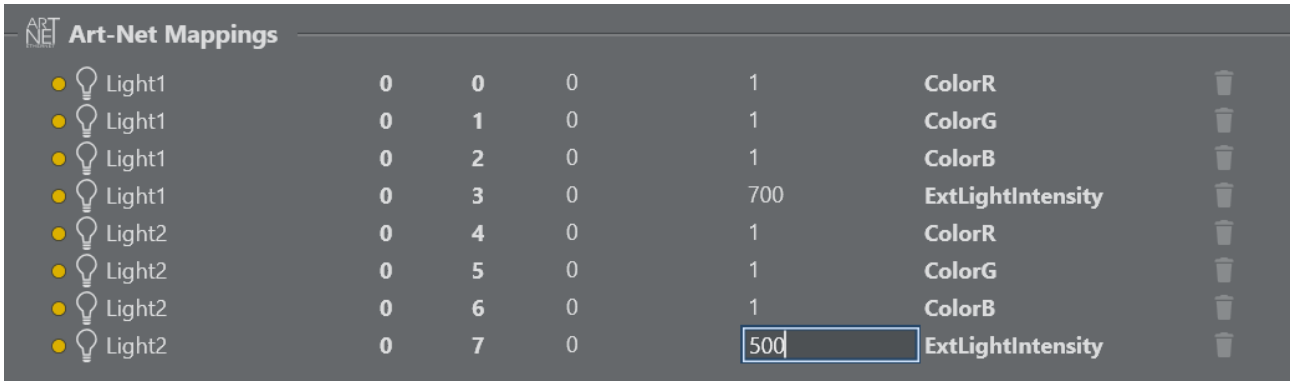
The overall DMX mapping can be viewed in the **Config Settings** panel under the **Shortcuts** section.

Copyright © 2025 Vizrt

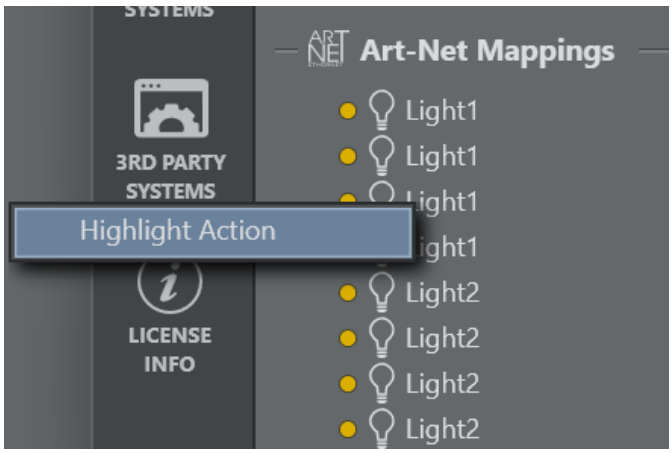


From here it is possible to remove a mapping and to re-adjust the minimum and maximum range mapping.

Copyright © 2025 Vizrt



By right clicking on the action the action can be highlighted on the action canvas.

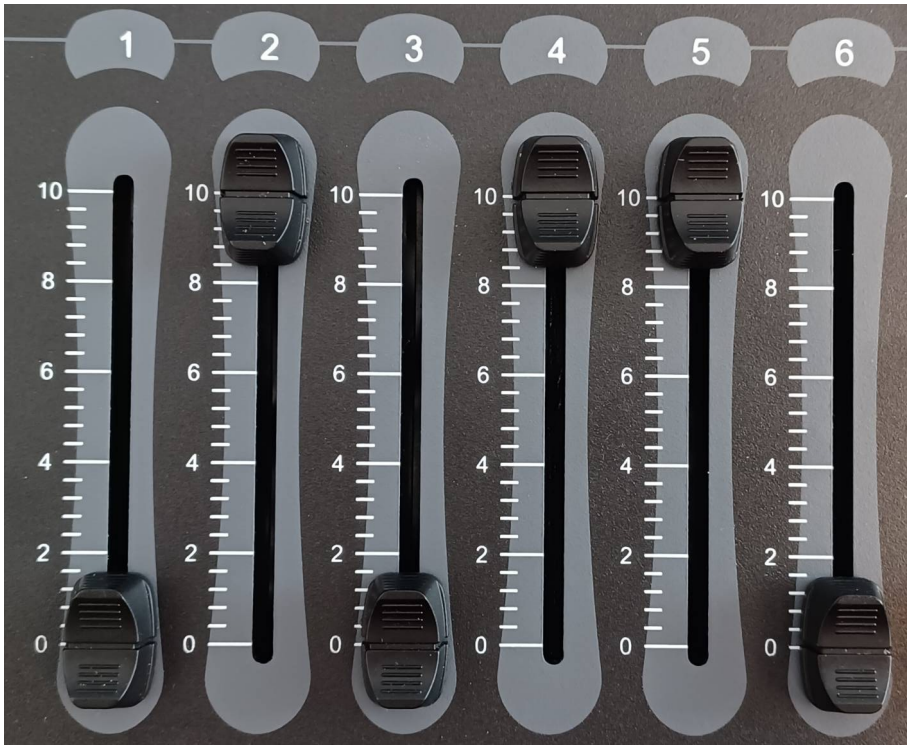


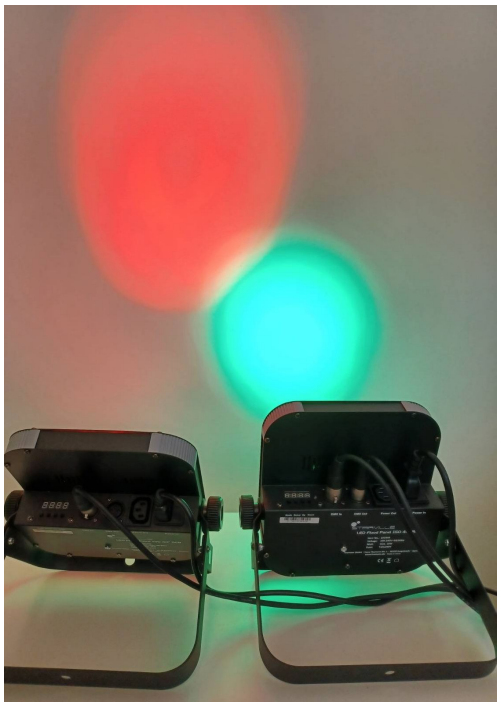
The rendered output of 2 virtual viz lights mapped through physical lights.



The physical lights controlled by a DMX light console where channel 0-3 are RGB of the first light, channel 4 is mapped to the intensity and with an offset of four channels.

The second light is mapped on sliders 5-9 of the console.





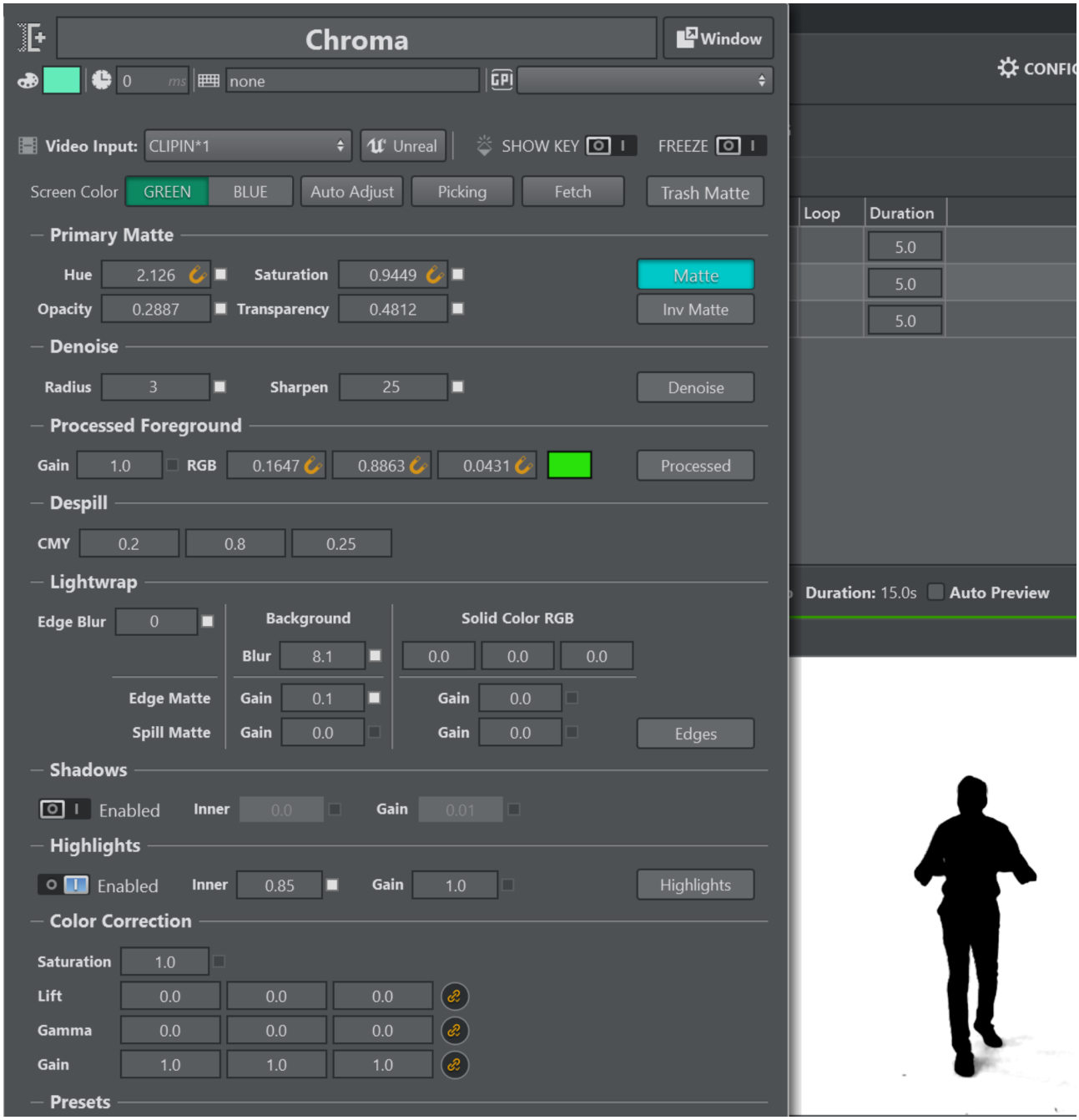
Copyright © 2025 Vizrt



Usage with Precision Key Action

The same way as in the Light Action, DMX signals can be used to control any floating point value in the Precision Key Action.

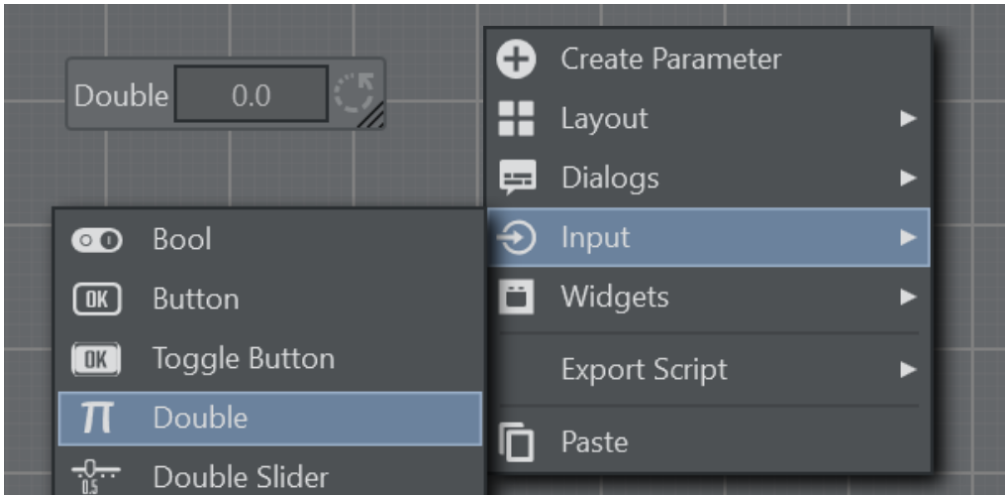
Copyright © 2025 Vizrt



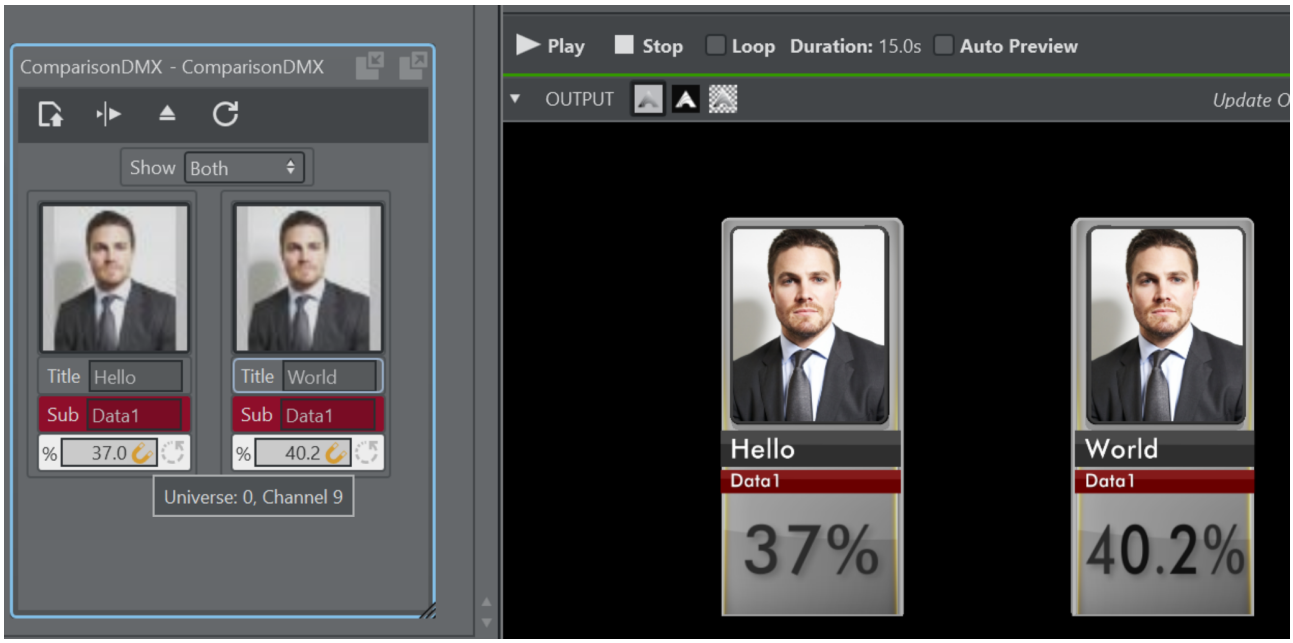
Usage with Template Actions

DMX signals can be used on double parameters only within Template Actions.

Copyright © 2025 Vizrt

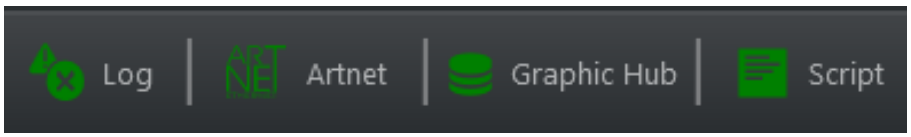


A sample of two double parameters representing a percentage connected to a DMX lighting control

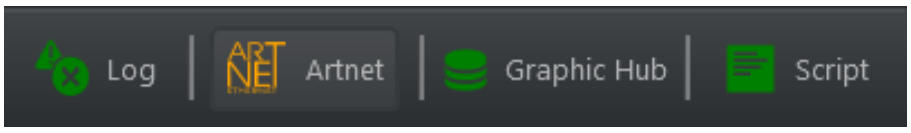


Enabling and Disabling DMX Events

On the bottom right status bar of the main window, you can cut-off all DMX events by clicking the button.



When the events are disabled, the icon is colored in orange. Click again to enable events.



Copyright © 2025 Vizrt

The events still arrive to the configuration wizard so you can still map DMX events to UI elements.

Scripting

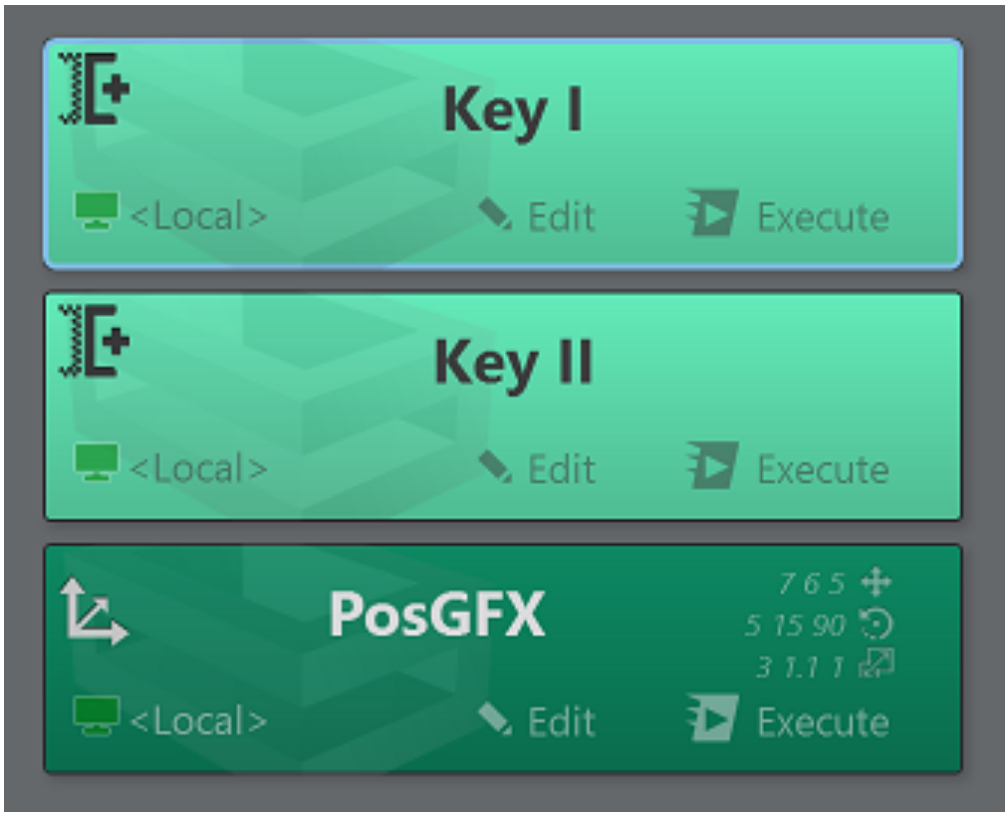
All DMX events can be intercepted and processed using the **OnDMXEvent** callback method. Additional methods **EnableDMX**, **DisableDMX** and **IsDMXEnabled** are available to enable and disable DMX events.

Copyright © 2025 Vizrt

5.8.2 Loupedeck

Getting Started

Start **Viz Arc** and then either select an existing Viz Arc project or create some chroma or transformation actions. It is sufficient to **select** any of those actions to modify its content. You can also edit the action and see the values changing when changed through the Loupedeck device.



Precision Keyer

On the Loupedeck device touch the **Viz Arc Control** button and in case a chroma action has been selected, the device shows with the first page of the chroma settings.

From here the **Hue**, **Saturation**, **Opacity** and **Transparency** can be changed using the dial buttons. The buttons on the top row change Viz Engine's preview output mode.

Copyright © 2025 Vizrt



Touch the right arrow to get to page two where settings related to the **denoise** can be set.

Copyright © 2025 Vizrt



Page three allows you change the **despill CMY** color and the **backing plate RGB** color.

Copyright © 2025 Vizrt



Page four allows you enable/disable shadows and highlights and change their respective parameters.

Copyright © 2025 Vizrt



Page five allows you to change the lightwrapping edge parameters.

Copyright © 2025 Vizrt



Page six allows to change more lightwrap parameters.

Copyright © 2025 Vizrt



Page seven allows you to manipulate the saturation and RGB lift.

Copyright © 2025 Vizrt



Page eight allows you to change gain RGB and gamma RGB.

Copyright © 2025 Vizrt



Transformations

Select a transformation action from the action canvas or from the map view.

The first page allows you to change **position** and **rotation** of the graphics element.

Copyright © 2025 Vizrt



The second page allows you to **scale** the graphics object. Select **Lock**, **Single** or **Proportional** for the scaling mode.

Copyright © 2025 Vizrt

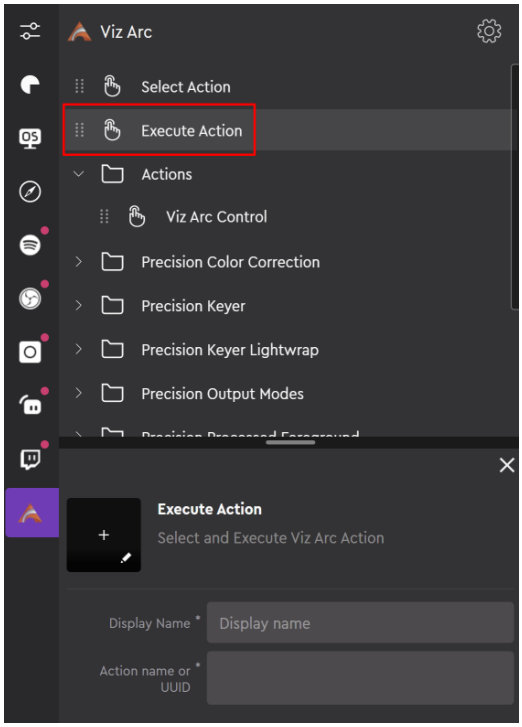


Other Functions

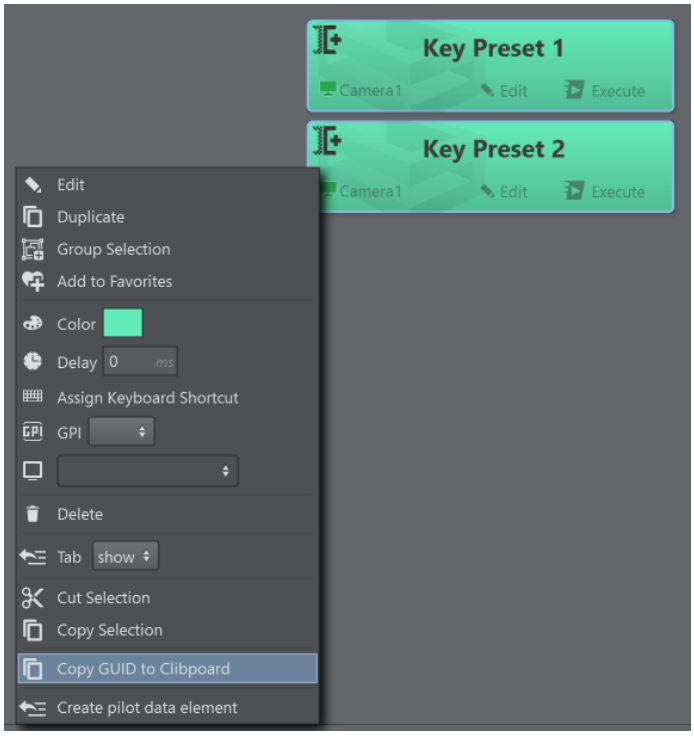
The **Execute Action** allows the selection and execution of a specific Viz Arc action from the Loupedeck device and is not limited to chroma or transformation actions.

Click on the **Execute Action** item to create a new command.

Copyright © 2025 Vizrt

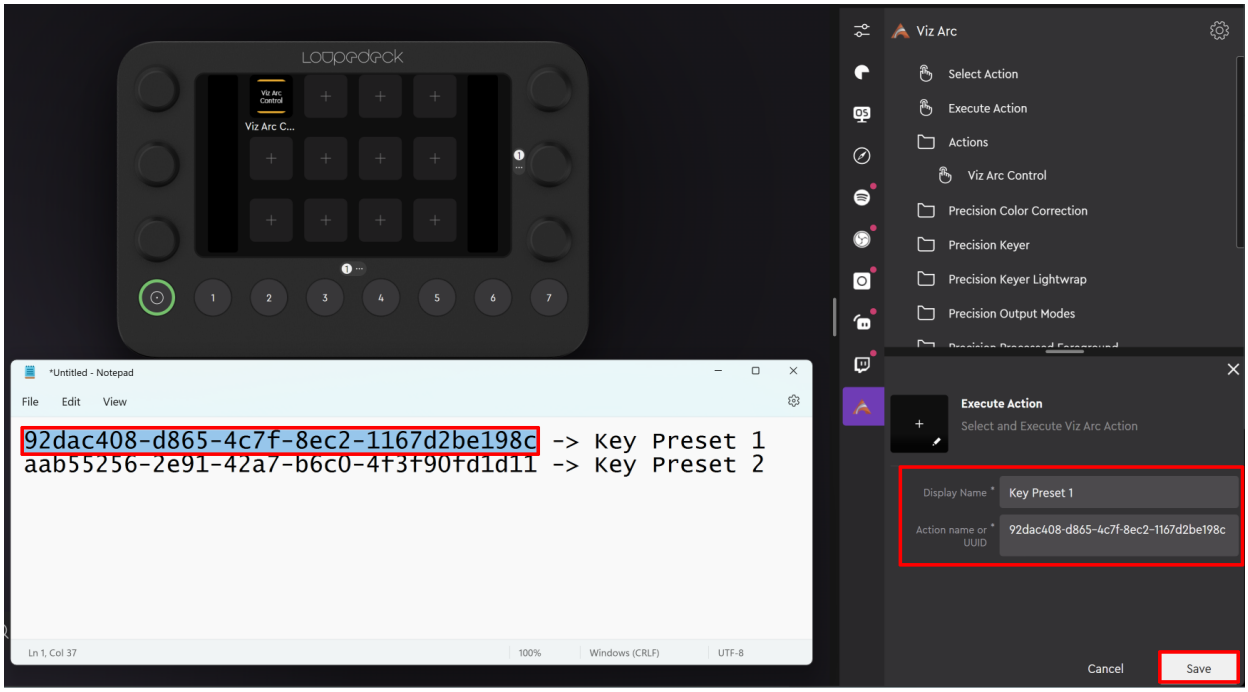


In Viz Arc, look for the actions you want to map on the Loupedeck device, select them, right click and select **Copy GUID to Clipboard**.



Copy the GUID's and names to notepad or similar and use either the name or the GUID and copy it over to the Loupedeck Execute Action Editor. Enter a name for the button and save the command.

Copyright © 2025 Vizrt



Under **Stored Viz Arc actions** you can find the previously defined actions. Drag them to any of the free buttons on the Loupedeck panel.



By touching the buttons, the action is automatically selected, highlighted and executed in Viz Arc. The action to be executed must be present on the currently opened project or the button has no effect.

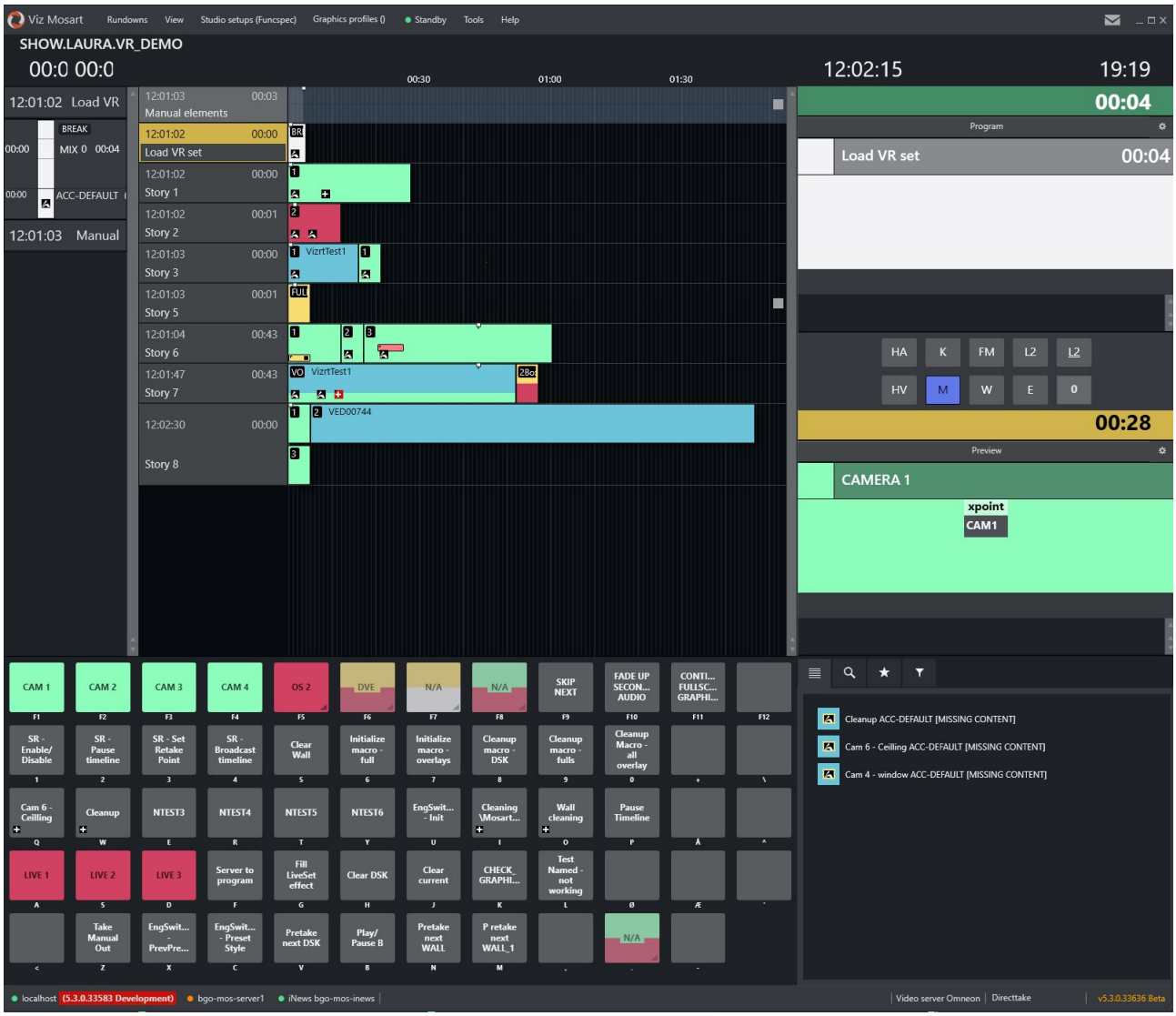
Limitations

Contemporary changes of values through the Loupedeck device and the Viz Arc action UI might result in unexpected behavior. The changes made through the Viz Arc UI are not instantly reflected on the Loupedeck device. You need to deselect and reselect the action in Viz Arc to push changes back to the external device.

Copyright © 2025 Vizrt

5.8.3 MOS Integration

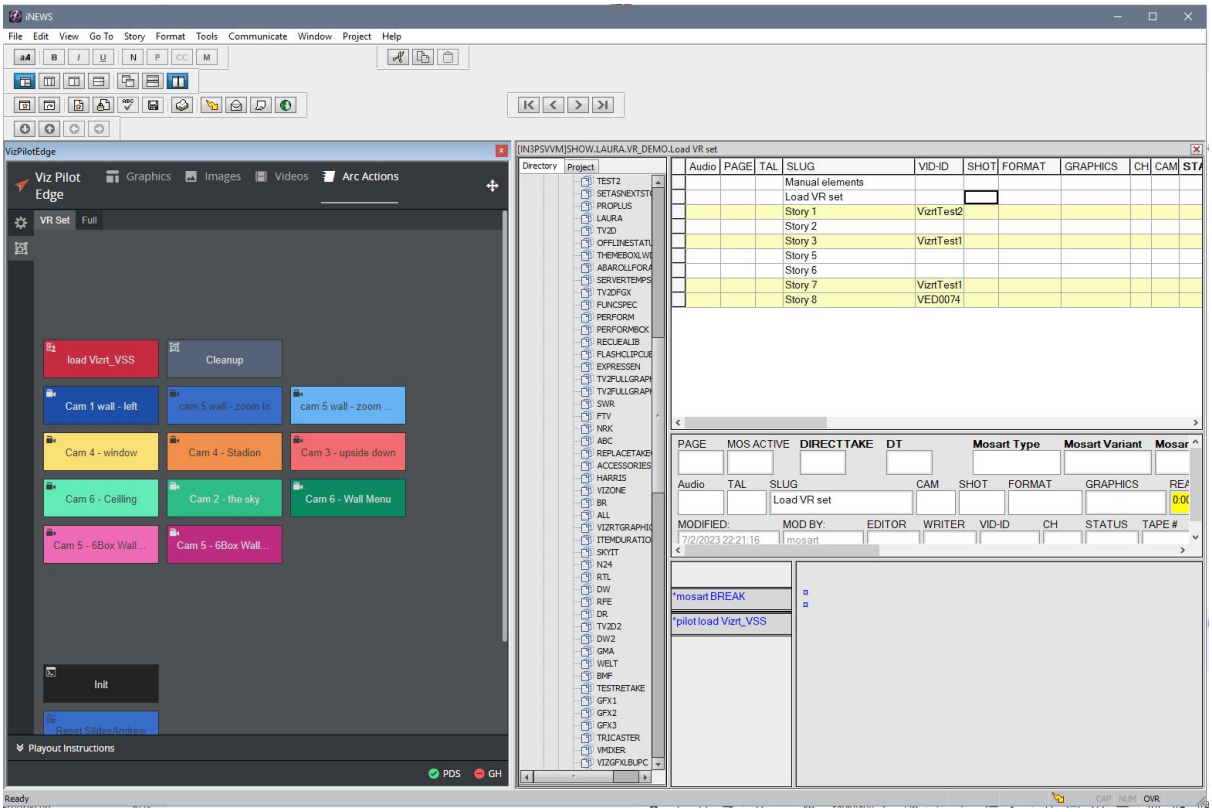
The MOS integration allows you to trigger single actions of any project from a newsroom rundown.



Prerequisites

- Installation of Viz Arc Web Service.
- A Viz Arc Rest license.
- A Viz Pilot Edge installation.

Copyright © 2025 Vizrt



- A newsroom system or Viz Mosart.

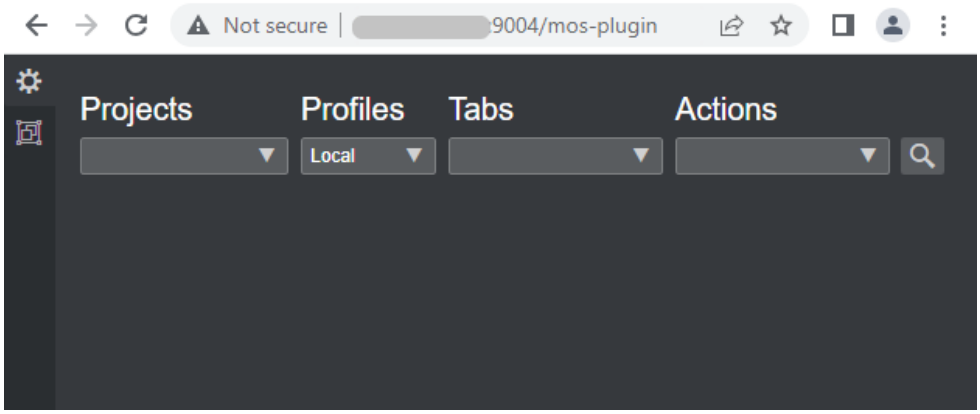
Check the Set-up

The Viz Arc **REST Web API** runs as a Windows service and does not need to be installed on the same machine where Viz Arc is running. If it runs on a different location it only has access to the projects stored on the Graphic Hub. It is recommended to also install Viz Arc, as the configuration is done through the Viz Arc configuration panel. It is important to configure the following settings: profiles, the Graphic Hub REST location and the Viz Arc REST Web API Port.

> **Test the Server:** You can test the server by entering the host name, the configured port (default `9004` for HTTP and `9005` for HTTPS) and the **mos-plugin** endpoint. For example, https://arc-server.my-company:9005/mos-plugin or for insecure connection http:// arc-server.my-company :9004/mos-plugin.

The server should respond as shown in the image below:

Copyright © 2025 Vizrt

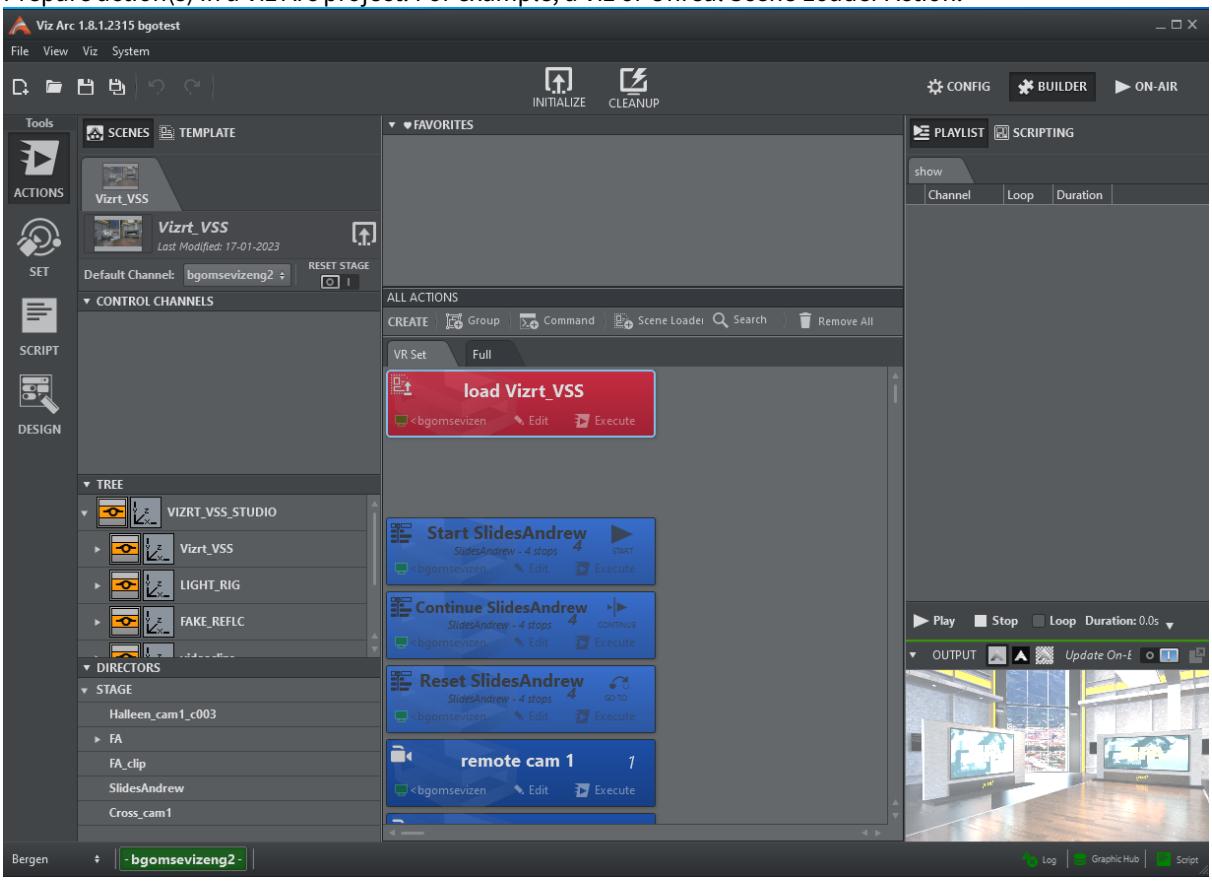


Note: Further configuration of the Database Parameters, such as *moseditor_arc_url*, *moseditor_arc_schema* and *moseditor_arc_label*, can be found in the section **Setup and Configuration** > **Viz Arc** of the Pilot Edge documentation.

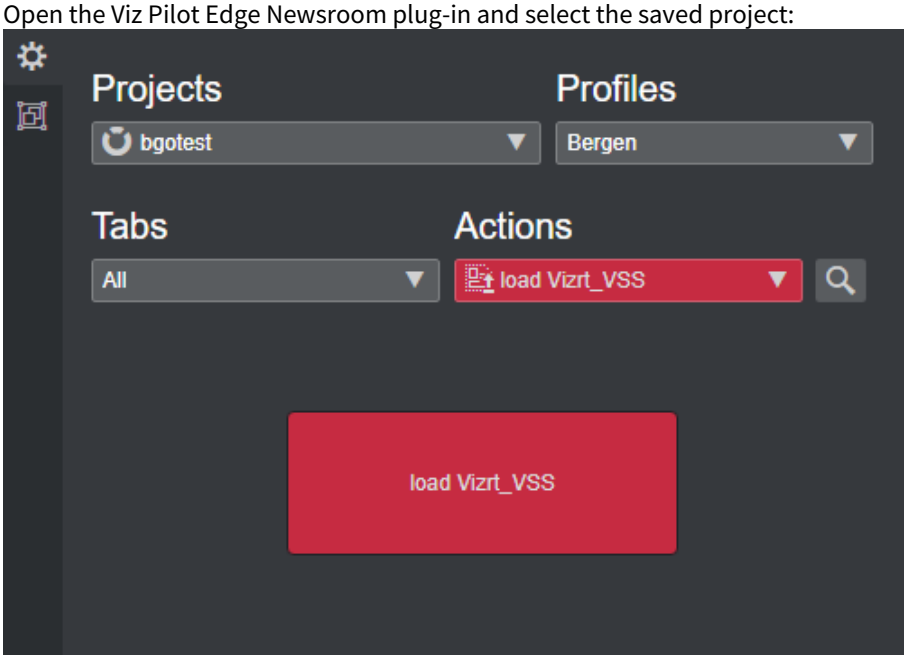
Note: Please refer to the Viz Mosart documentation for further configuration of the MSE Host for Viz Arc.

Usage Workflow

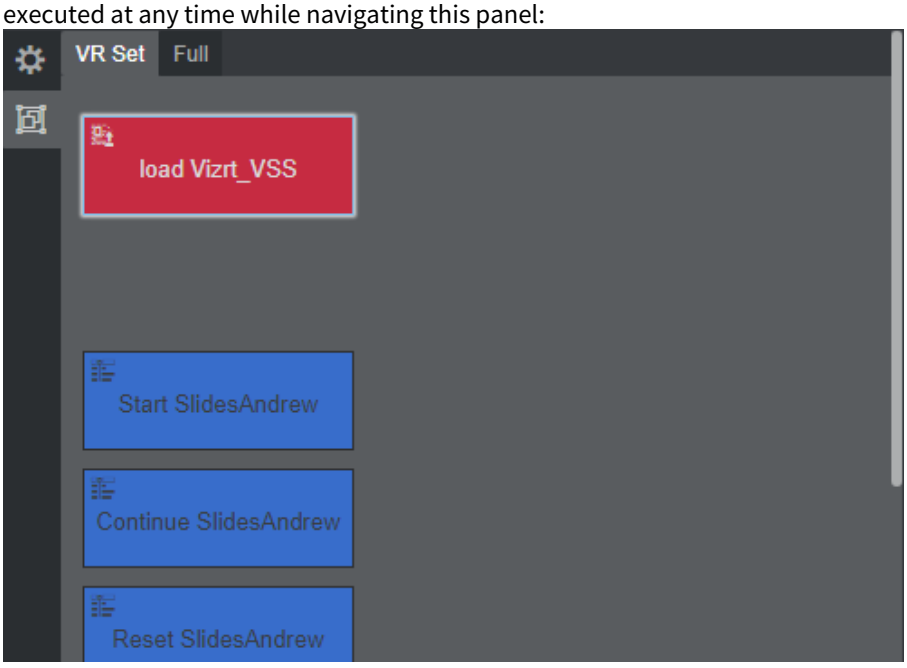
1. Prepare action(s) in a Viz Arc project. For example, a Viz or Unreal Scene Loader Action:



2. Save the project (preferably on Graphic Hub).
3. Open the Viz Pilot Edge Newsroom plug-in and select the saved project:

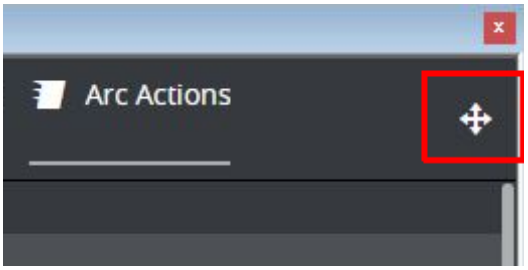


4. Select the project from the **Projects** dropdown, then select the profile to be used for the playout from the **Profiles** dropdown.
5. Finally, select the action to be played out from the **Actions** tab. Select **All** or any specific Viz Arc Tab to filter the actions. You can also navigate the canvas view and select the action by clicking on it. The action is NOT executed at any time while navigating this panel:



6. Drag the action from the Viz Pilot Newsroom plug-in into the newsroom system rundown.

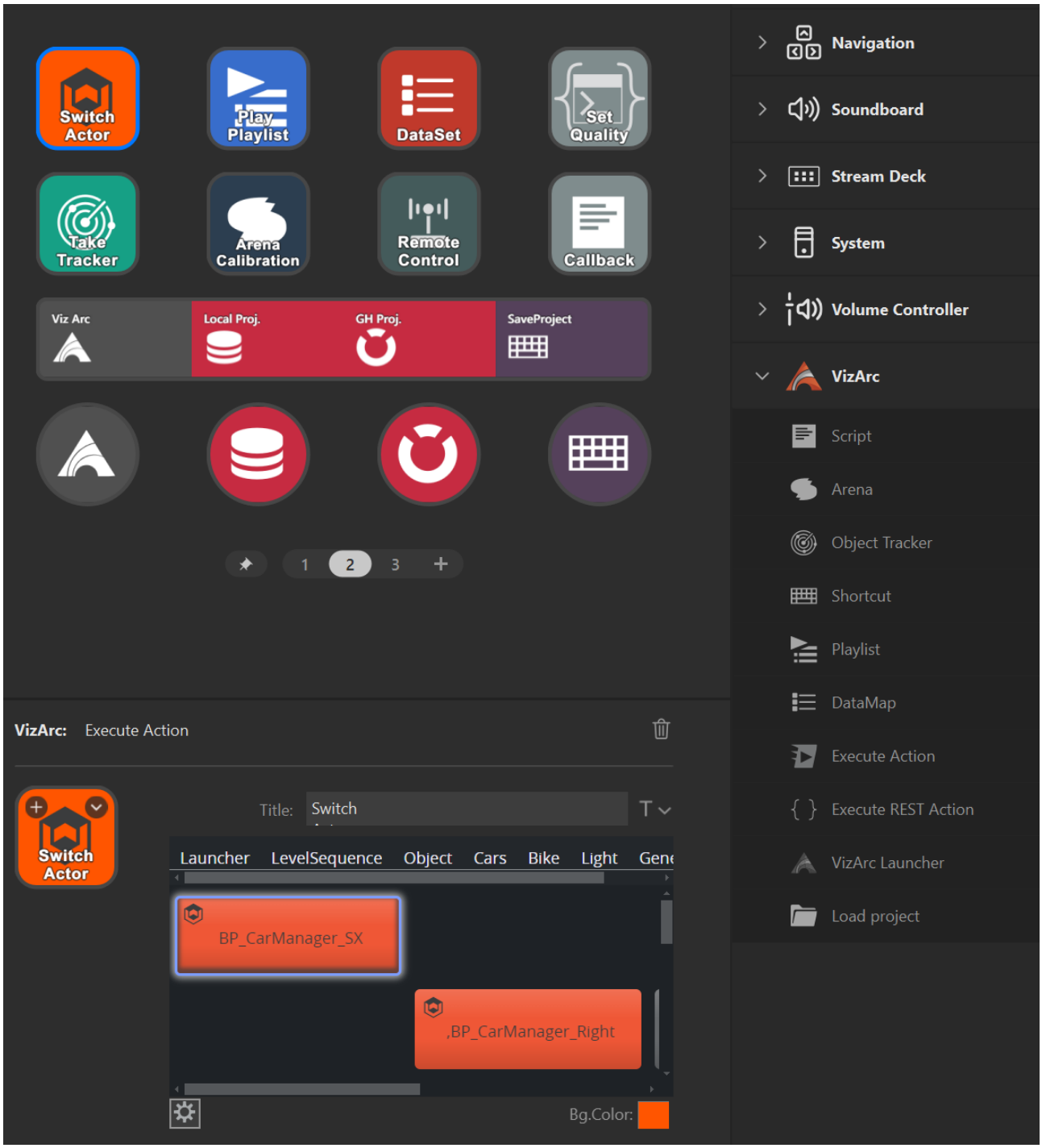
Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

5.8.4 Stream Deck

This section describes the various Stream Deck plug-ins for Elgato's.

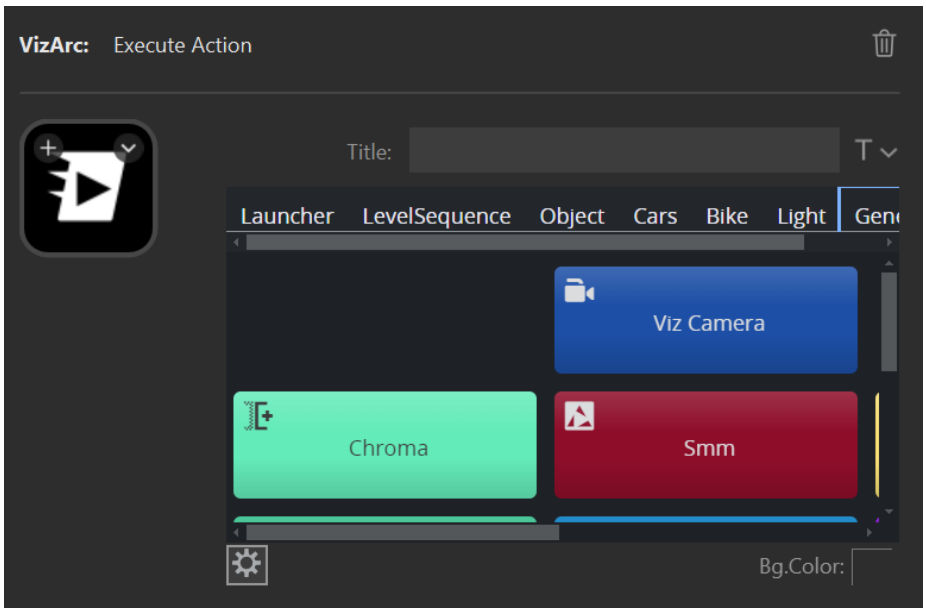


Note: Most plug-in actions require Viz Arc to be running.

Copyright © 2025 Vizrt

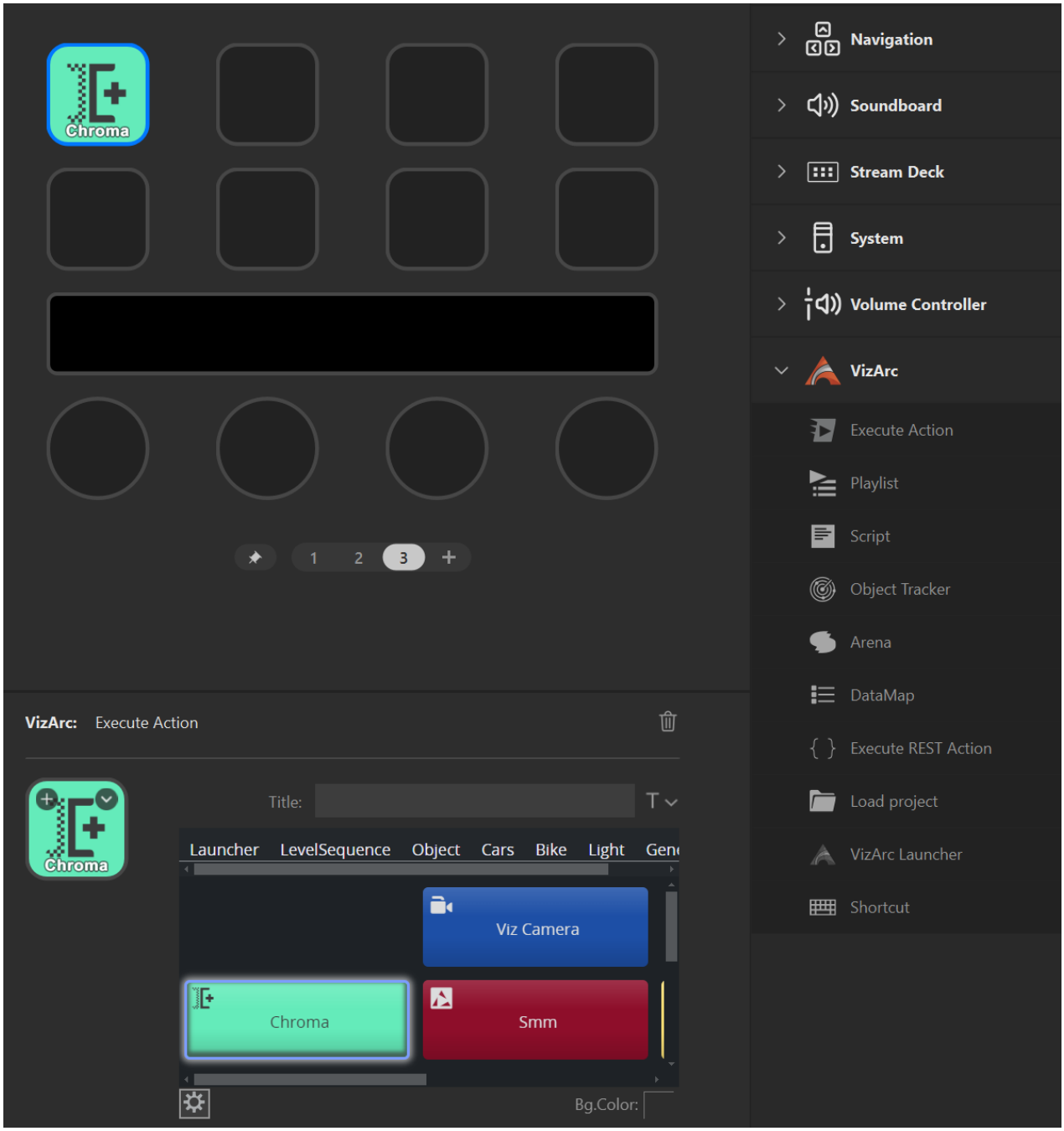
Execute Action

Drag and drop the **Execute Action** item to any of the Stream Deck buttons. The action lets you select an Action from the currently loaded project by selecting any of the actions.



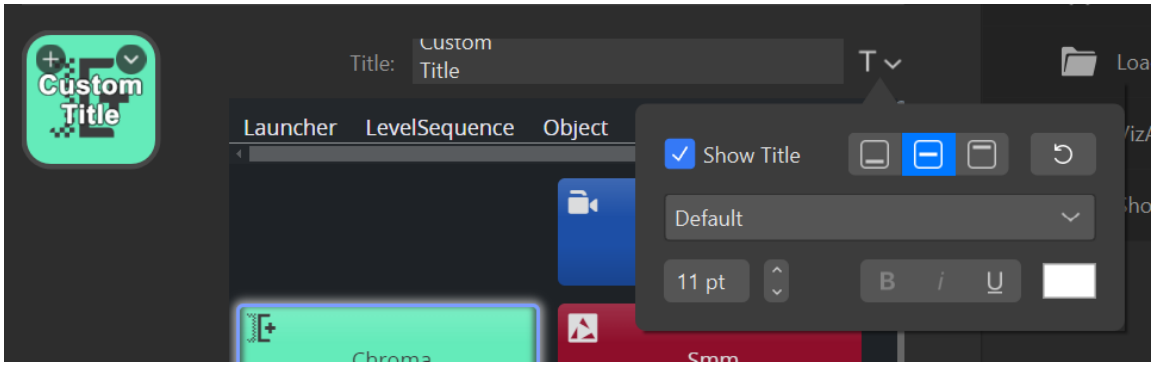
Note that the action is **NOT** executed when selecting it by clicking on it. Once the action has been selected, the button shows up with the action icon and action name.

Copyright © 2025 Vizrt

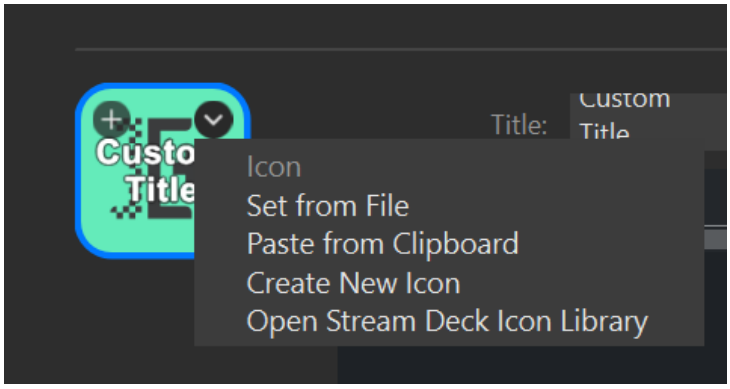


The name can be customized through the **Title** property editor.

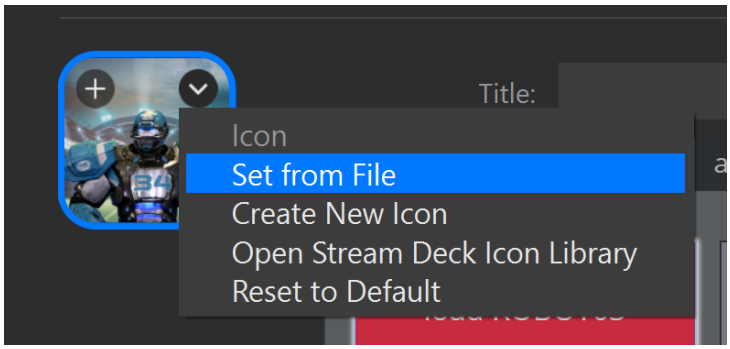
Copyright © 2025 Vizrt



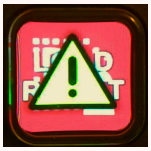
The background can be customized as you please using a wide range of sources and editors.



And example loading an icon from a *.jpg* file.



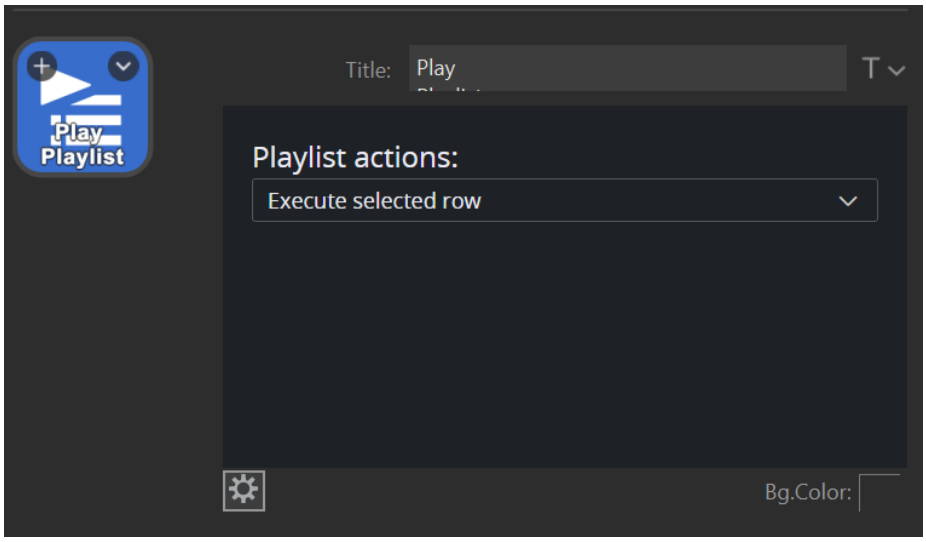
When executing: Once the Viz Arc Action to be executed has been selected, pushing the button executes the action only if Viz Arc is running and the project from which the action originates is currently opened. If the action cannot be found or Viz Arc is not running, a small triangle shows indicating that the execution did not succeed.



Copyright © 2025 Vizrt

Playlist

The Playlist action lets you control various aspects of Viz Arc's playlist.



Execute Selected Playlist Row: Plays the currently selected playlist element.

- **Execute Selected Playlist Row And Next:** Plays the currently selected playlist element and advances to the next playable playlist element.
- **Preview Selected Playlist Row:** Previews the selected playlist element.
- **Go to First:** Selects the first playlist element.
- **Play Current Playlist:** Plays the currently selected playlist from the selected item on.
- **Stop Current Playlist:** Stops the currently selected playlist.
- **Play Playlist By Name:** Selects and plays the playlist by the specified name. If the name is ambiguous, it plays the first occurrence.
- **Play Playlist By Index:** Selects and plays the playlist specified by the given index, where 0 is the first playlist.
- **Change Active Tab:** Selects the tab by the specified name.

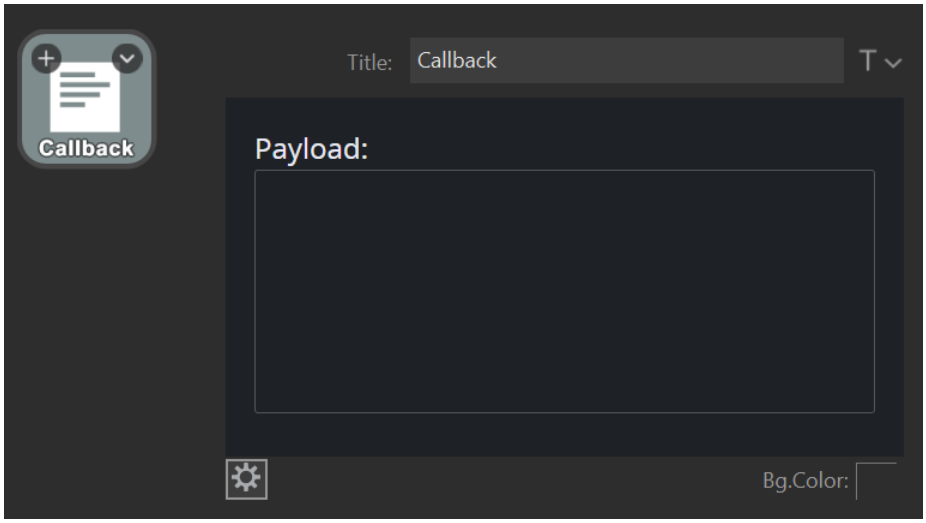
Refer to the playlist to learn more about Viz Arc's playlist functionality.

Script Callback

The Script Callback action triggers Viz Arc script functions that may be defined in the global script or a template.

The **Payload** can be any string passed to the script method.

Copyright © 2025 Vizrt



A sample Viz Arc script might look like this:

Viz Arc Script Sample

```
function printSDEvenyInfo(sdEvent)
{
    // print all the available information for a StreamDeck event
    Console.WriteLine("StreamDeck Event Info:")
    Console.WriteLine("Device Index: " + sdEvent.DeviceIndex)
    Console.WriteLine("Device ID:    " + sdEvent.Id )
    Console.WriteLine("Column:       " + sdEvent.XKey )
    Console.WriteLine("Row:          " + sdEvent.YKey )
    Console.WriteLine("Payload:      " + sdEvent.Payload )
}

Global.OnStreamDeckKeyDown = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)

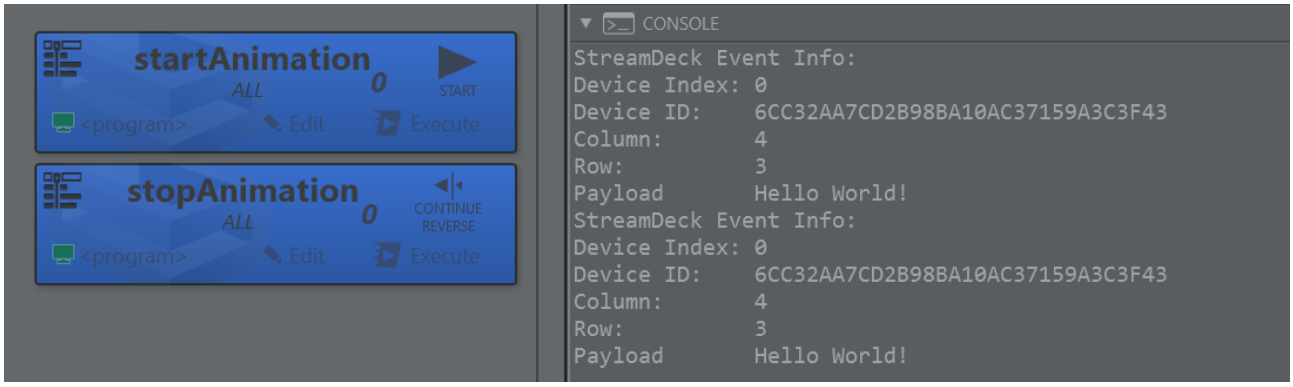
    // start an animation when StreamDeck key is down
    GetAction("startAnimation").Execute()
}

Global.OnStreamDeckKeyUp = function (sdEvent)
{
    printSDEvenyInfo(sdEvent)

    // stop an animation when SrreamDeck key lifts up again
    GetAction("stopAnimation").Execute()
}
```

The script above executes the startAnimation action when the Stream Deck button is pressed and the stopAnimation action when the button is released.

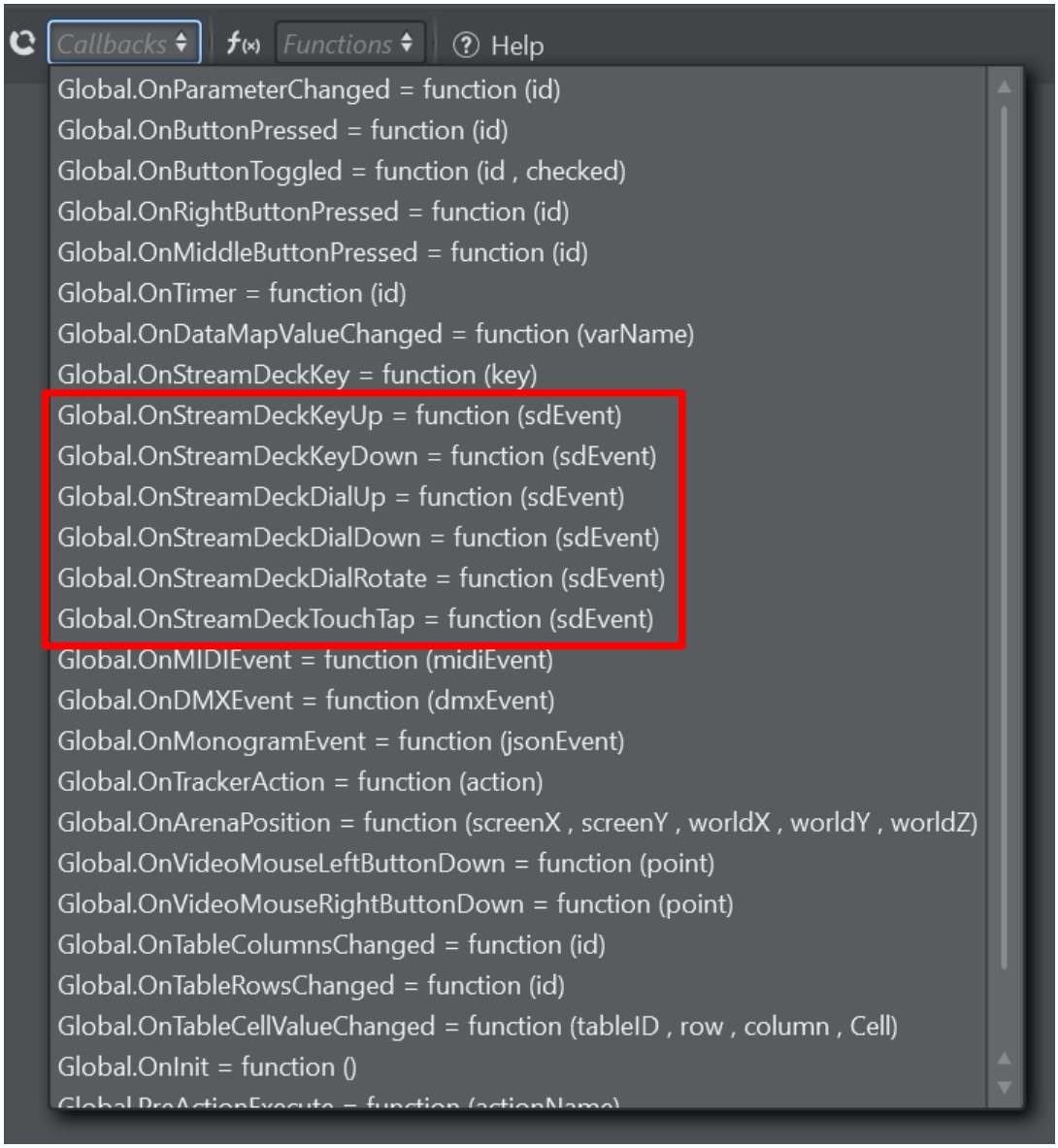
Copyright © 2025 Vizrt



The available script callbacks are:

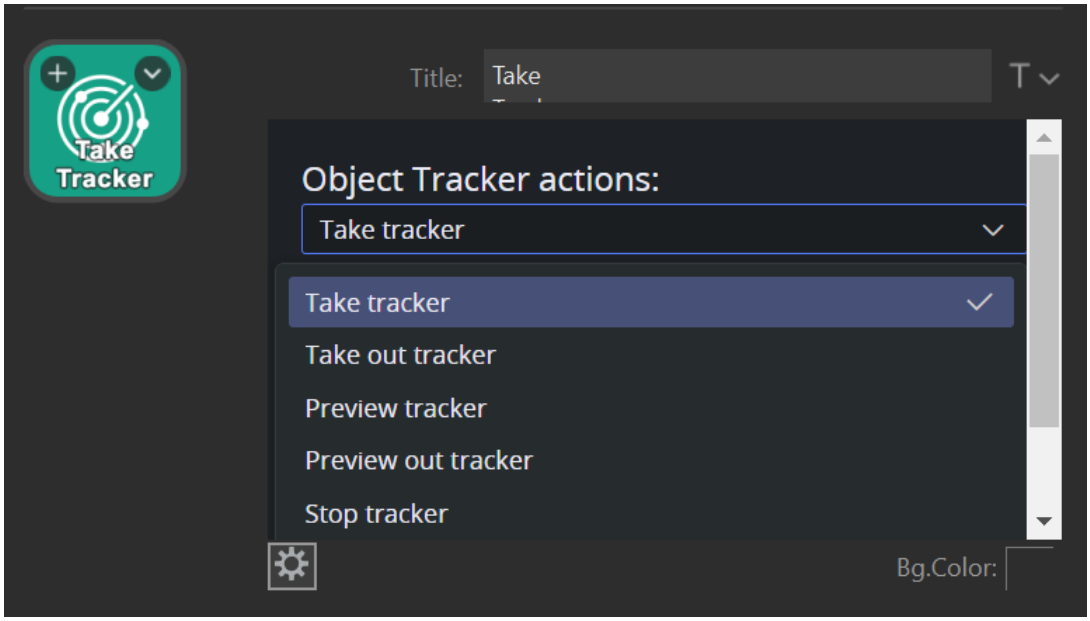
- **OnStreamDeckKeyDown:** Invoked when the key is pressed.
- **OnStreamDeckKeyUp:** Invoked when the key is released.
- **OnStreamDeckDialUp:** Invoked when a dial is pressed.
- **OnStreamDeckDialDown:** Invoked when a dial is released.
- **OnStreamDeckDialRotate:** Invoked when a dial gets rotated.
- **OnStreamDeckTouchTap:** Invoked when a user tapped a touch area button.

Copyright © 2025 Vizrt



Object Tracker

The Object Tracker action lets you execute Object Tracker related macros.



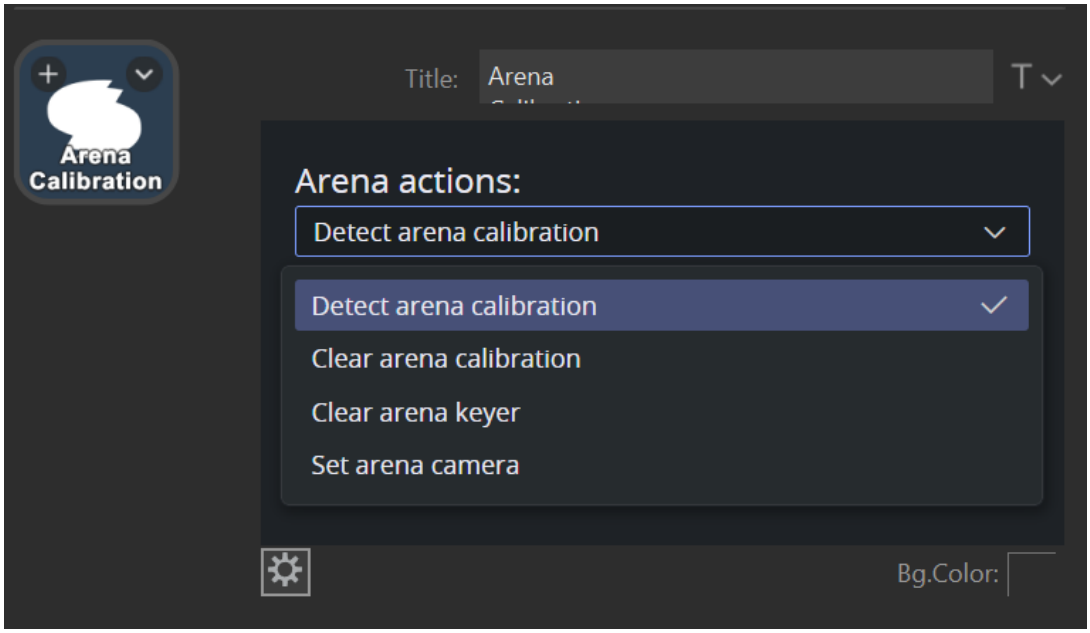
- **Take Tracker:** Takes all active trackers On Air.
- **Take Out Tracker:** Takes all active trackers Off Air.
- **Preview Tracker:** Previews all active trackers.
- **Preview Out Tracker:** Previews out all active trackers.
- **Stop Tracker:** Stops all active trackers.
- **Set Active Tracker:** Selects the currently active tracker by index, starting at 1.
- **Reset Pointer Offset:** Resets all pointer offsets.

Refer to the Viz Object Tracker manual for more information.

Arena

The Arena action lets you execute Arena related macros.

Copyright © 2025 Vizrt

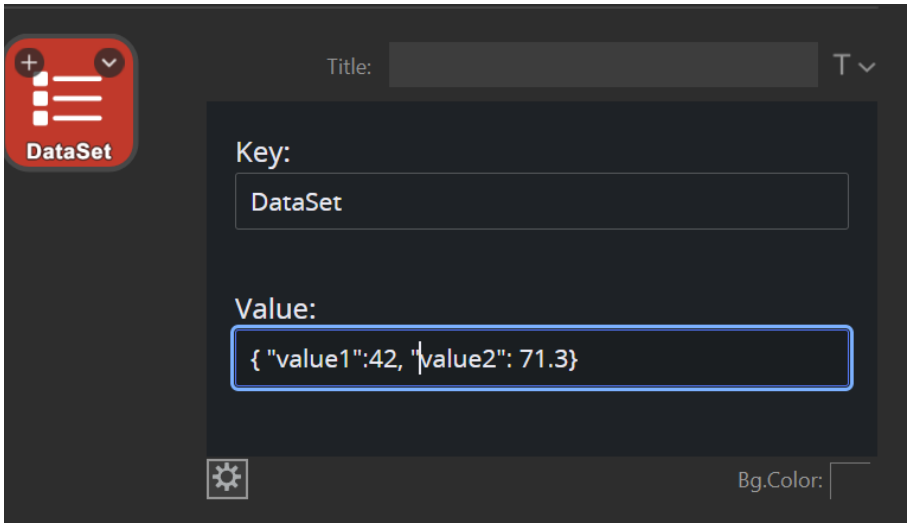


- **Detect Arena Calibration:** Detects the calibration of the current image, equivalent of pressing the shortcut **D** in Viz Arena.
- **Clear Arena Calibration:** Clears the calibration, equivalent of pressing the **BACKSPACE** button in Viz Arena.
- **Clear Arena Keyer:** Clears the keyer, equivalent of pressing **C**.
- **Set Arena Camera:** Selects the current camera for both Viz Arena preview and program feeds.

Refer to the Viz Arena Integration documentation for more information.

DataMap

The DataMap action allows you to set a Viz Arc DataMap variable.

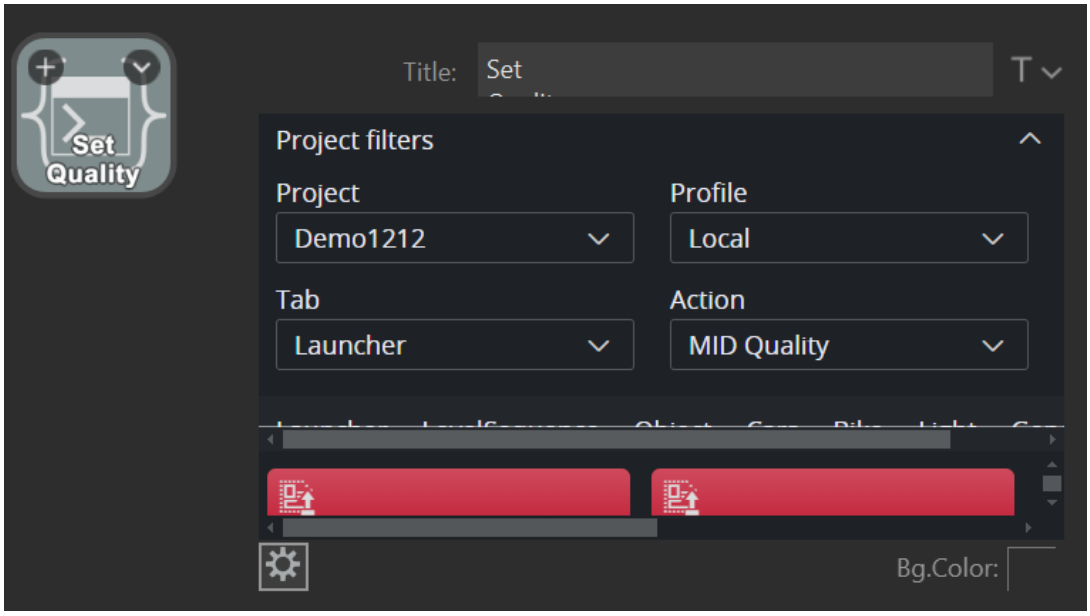


Copyright © 2025 Vizrt

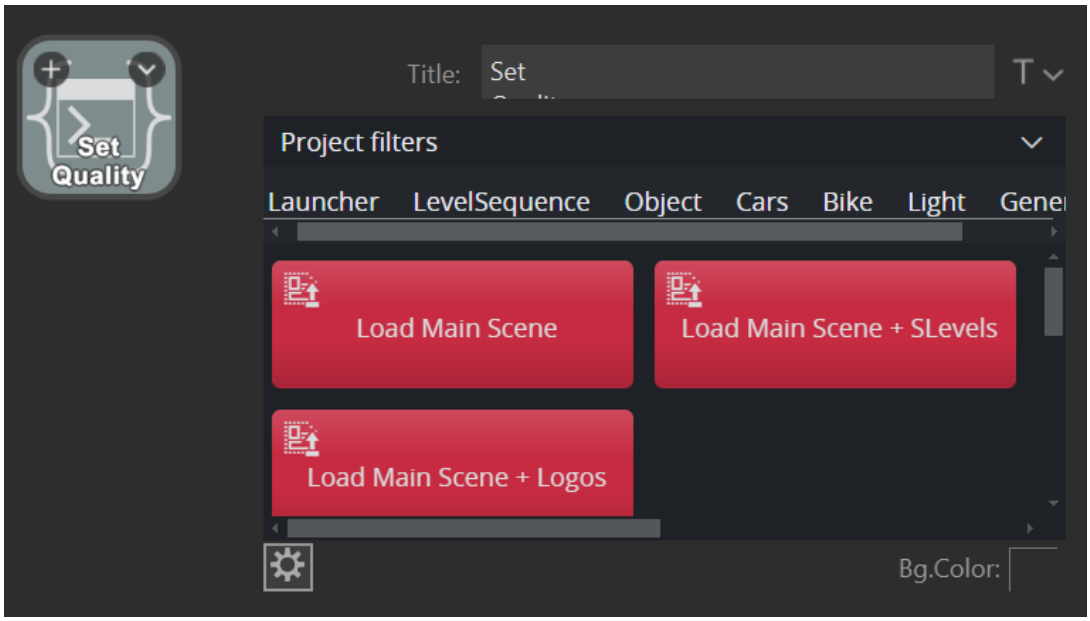
Execute REST Action

The Execute REST Action allows you to launch Viz Arc actions using the REST service which does not require Viz Arc to be running.

You need to specify a project, an action and a profile on which to execute the action.



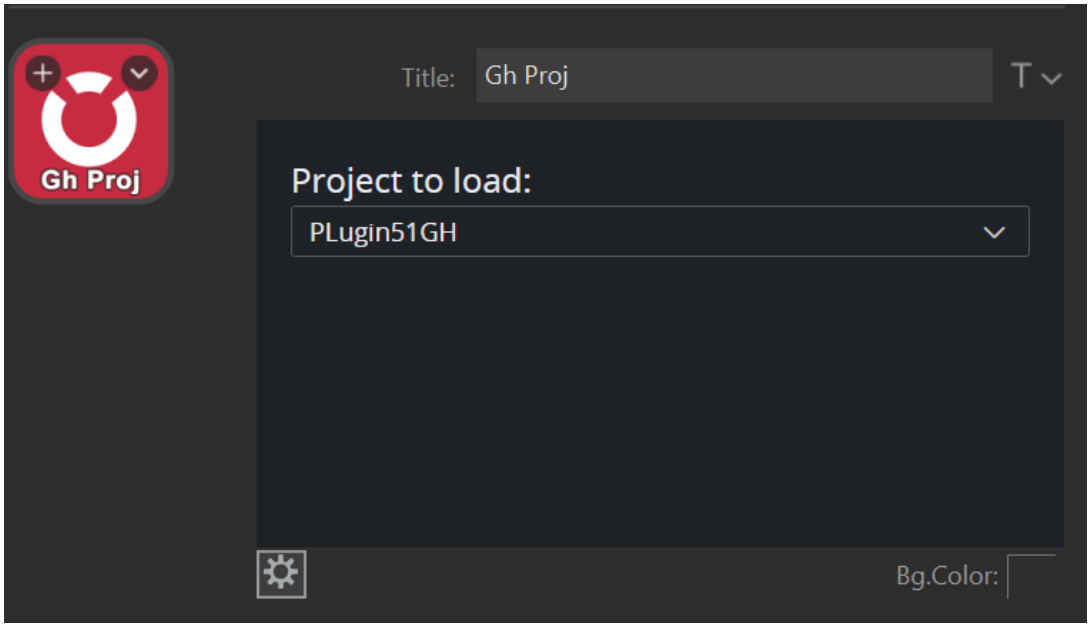
You can also navigate the action canvas and tabs to select an action.



Load Project

The Project Loader allows to load a project from Graphic Hub or a local file based project. Select a project to be loaded from the drop down menu.

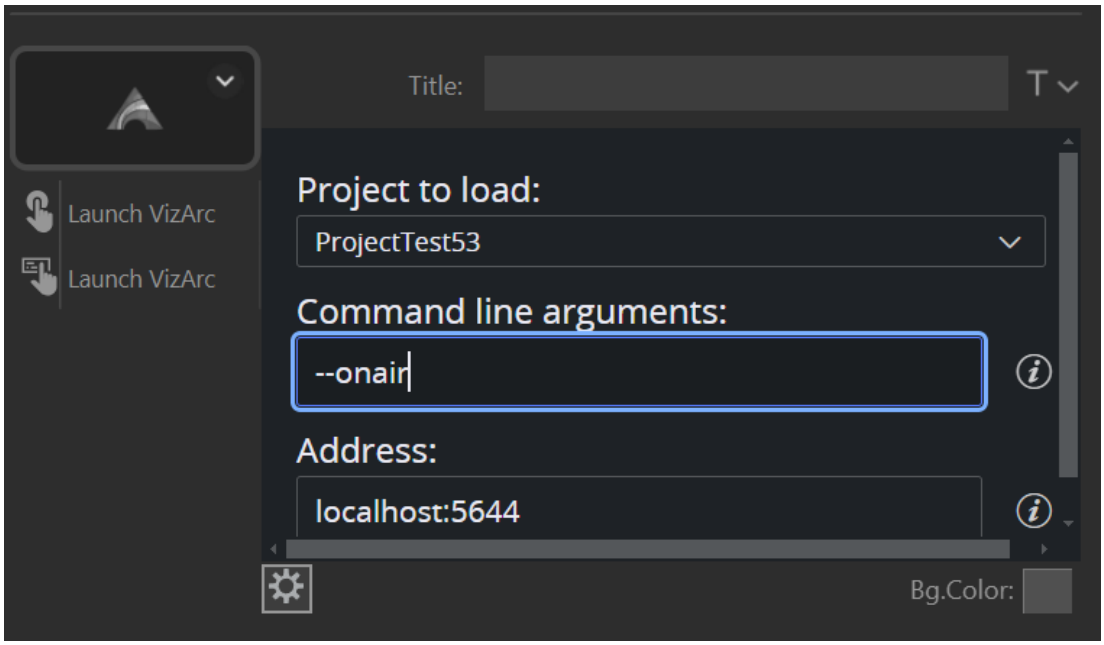
Copyright © 2025 Vizrt



Viz Arc Launcher

With the Viz Arc Launcher, you can start Viz Arc in case it is not running with a specific project with optional command line arguments.

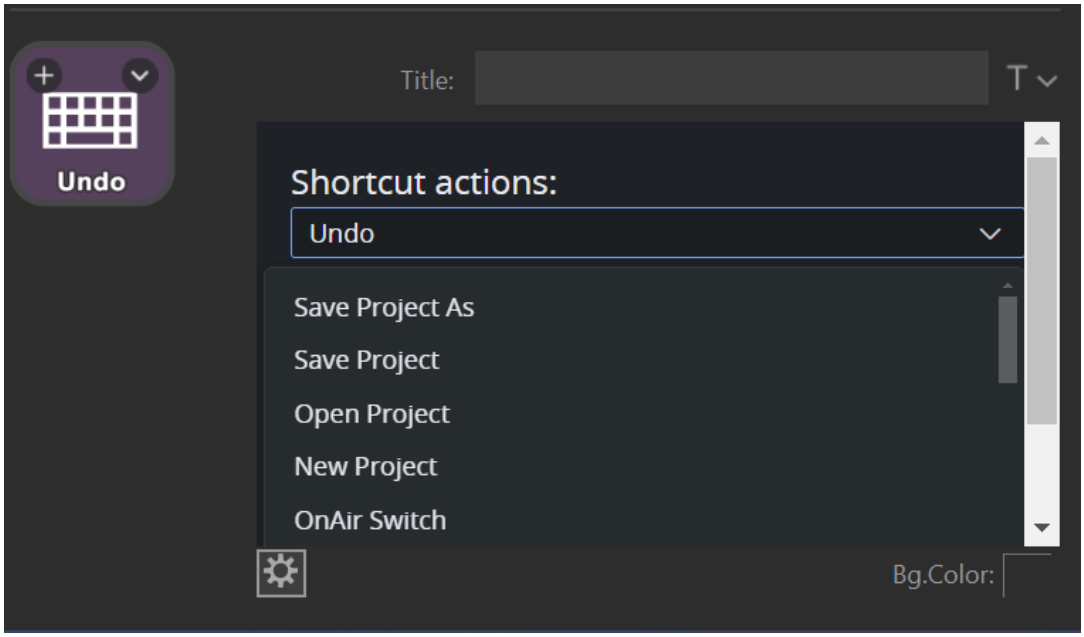
This plugin relies on an installed Viz Arc Unreal Loader service. Change the **Launcher address** to a different host/port if the Unreal Launcher and Viz Arc are installed on a remote host.



Shortcut

The Shortcut action lets you execute Viz Arc's shortcuts.

Copyright © 2025 Vizrt



Refer to Keyboard Shortcuts for more information.

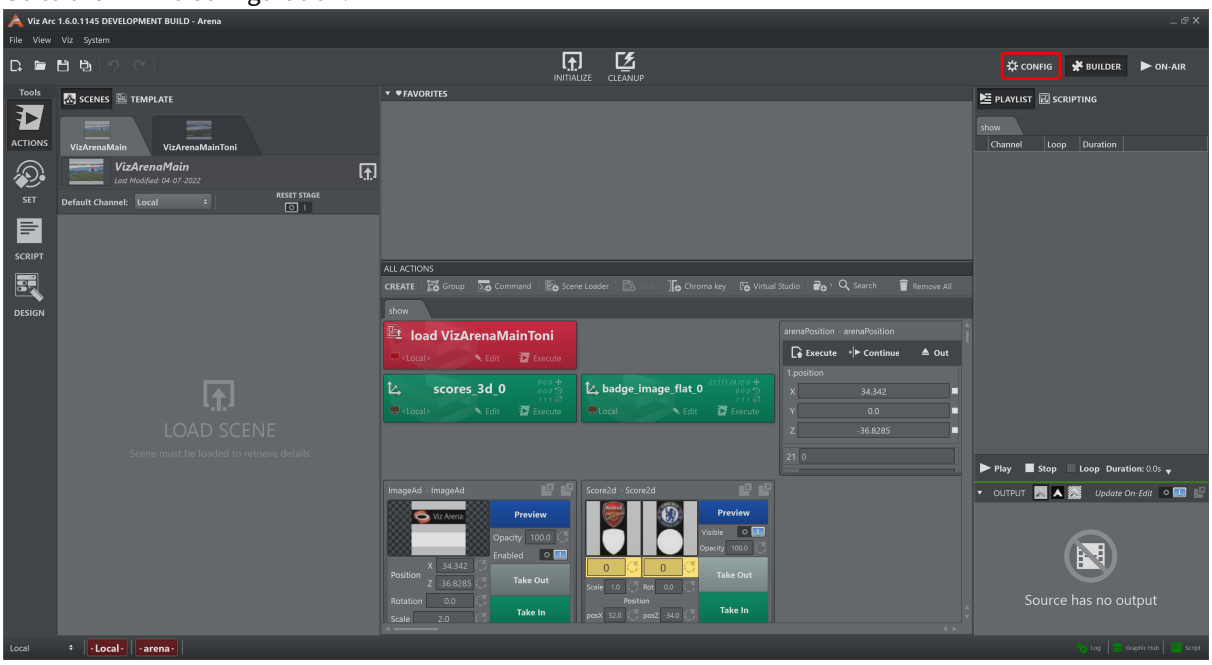
Copyright © 2025 Vizrt

5.8.5 Viz Arena

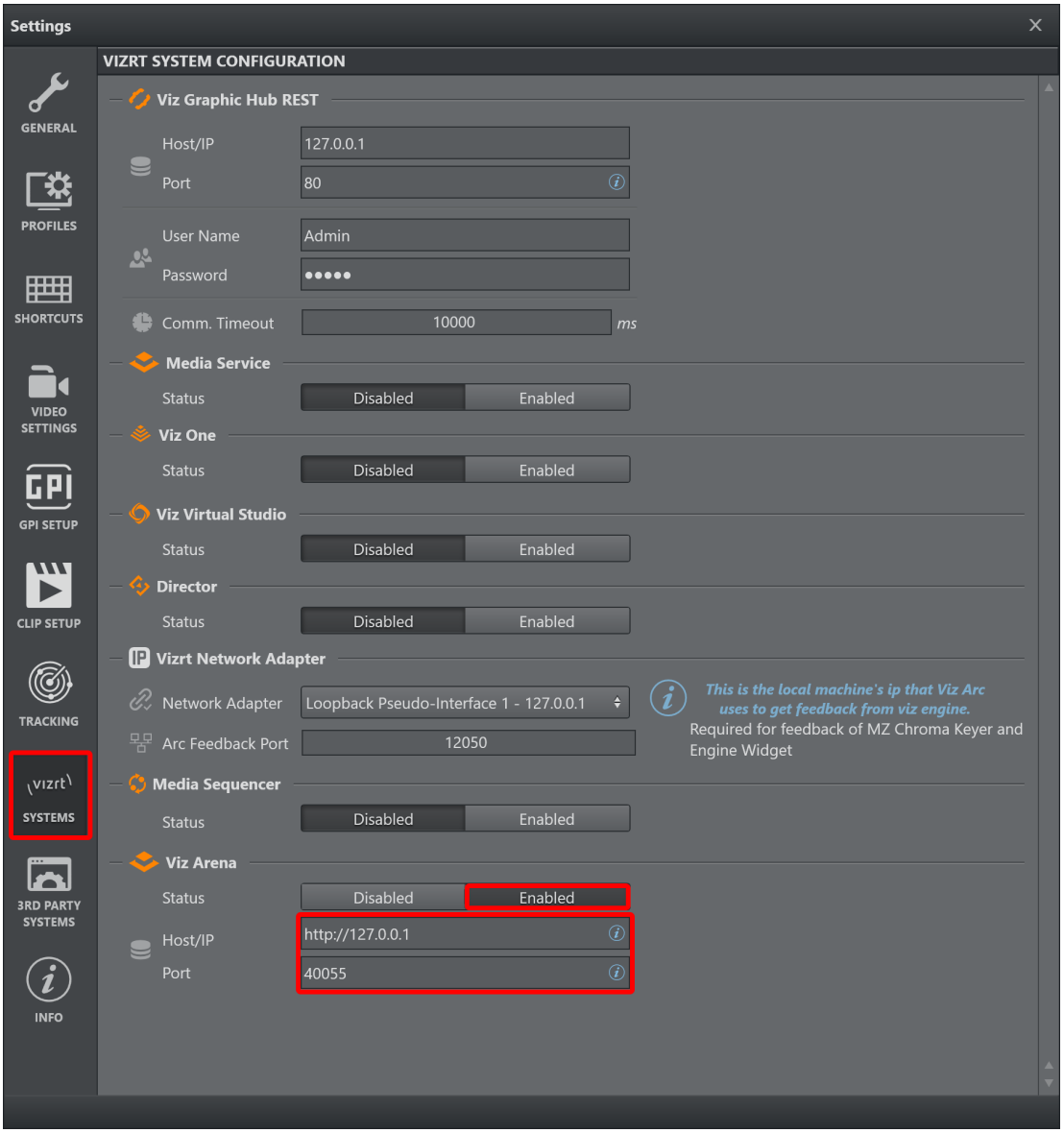
Configuration

First of all you have to to enable the Viz Arena panel:

- Go to the Viz Arc Configuration.

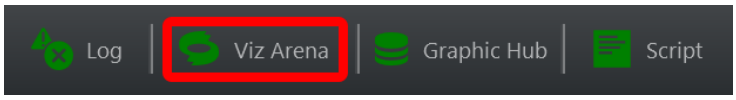


- Select the Vizrt System tab and press **Enabled** on Viz Arena.



- Insert the correct IP address of the workstation where Viz Arena resides and the port of the API server.

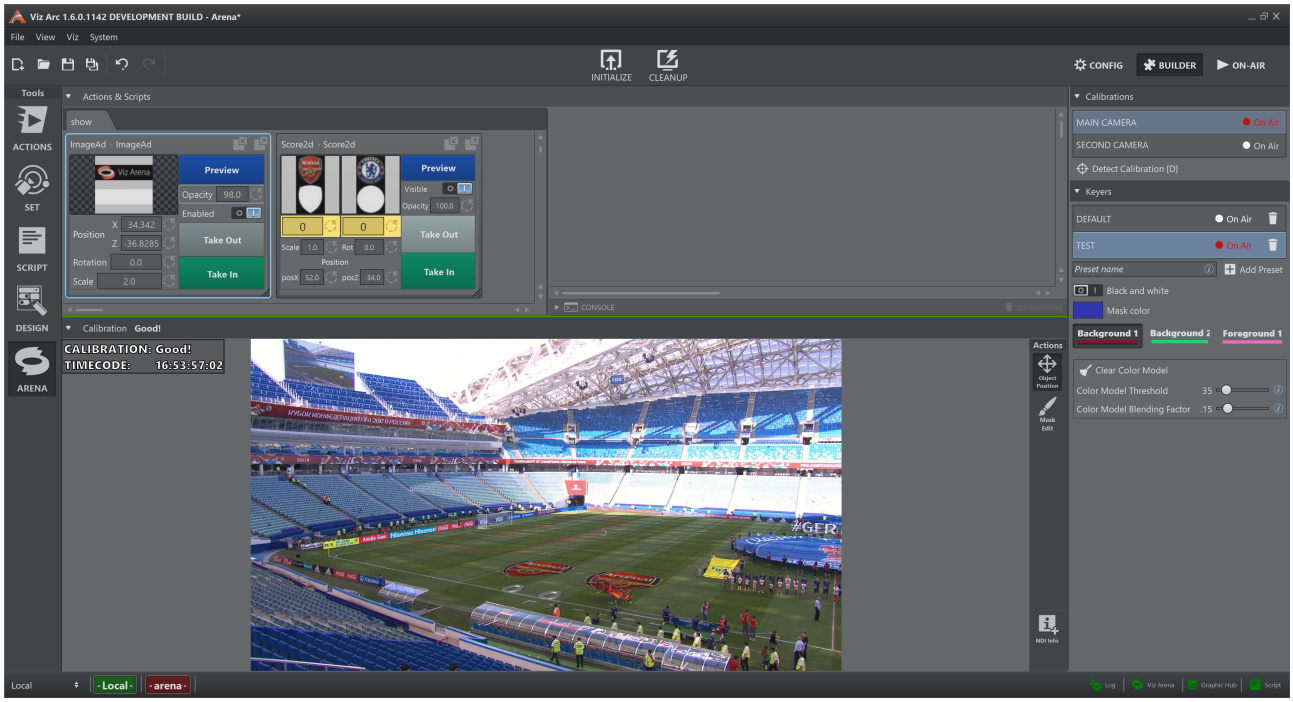
The Viz Arena status icon is located in the status bar (on the bottom right).



It turns green after a while if the IP address and port inserted are correct, otherwise it remains red.

Now it is available to press the Arena button on the "Tool toolbar" (on the leftmost portion of the window), the interface looks similar to this:

Copyright © 2025 Vizrt

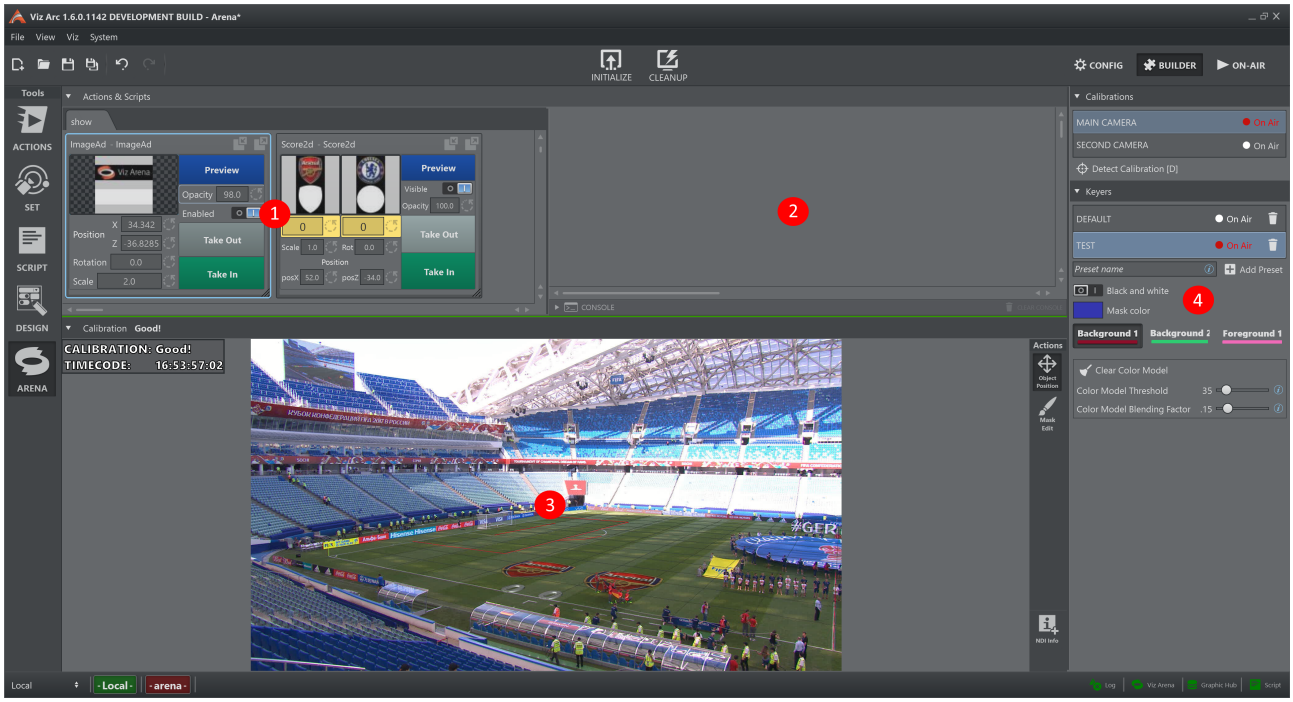


Features

The Arena tab presents multiple features:

- Actions panel (1), see Actions View for further information
- Script panel (2), see Script View for further information
- Output view panel (3)
- Settings panel (4)

Copyright © 2025 Vizrt



Output View

The Output View shows the NDI stream, both video and metadata, provided by Viz Arena. Since it is very important to see any changes of the calibration status, at the very top of the view there is line that show its current value:

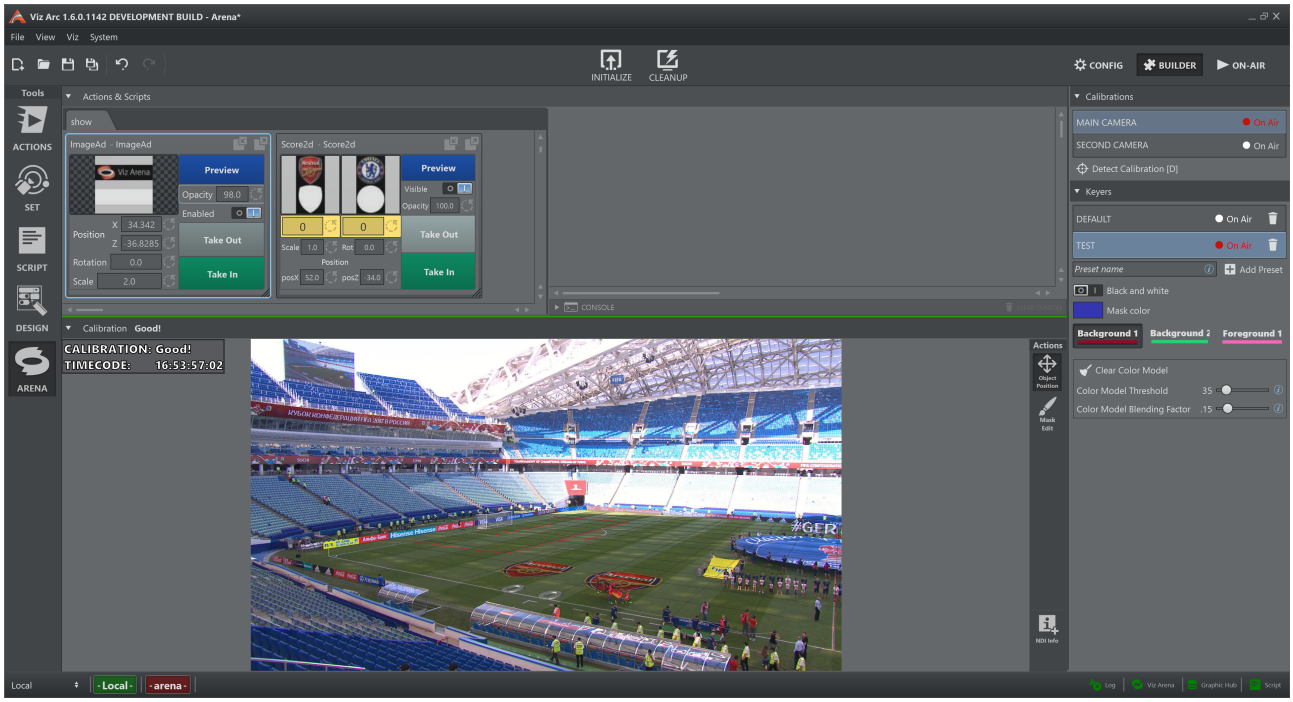
- **Green** if the calibration is good.
- **Red** if the calibration is lost.

Actions

Object Position

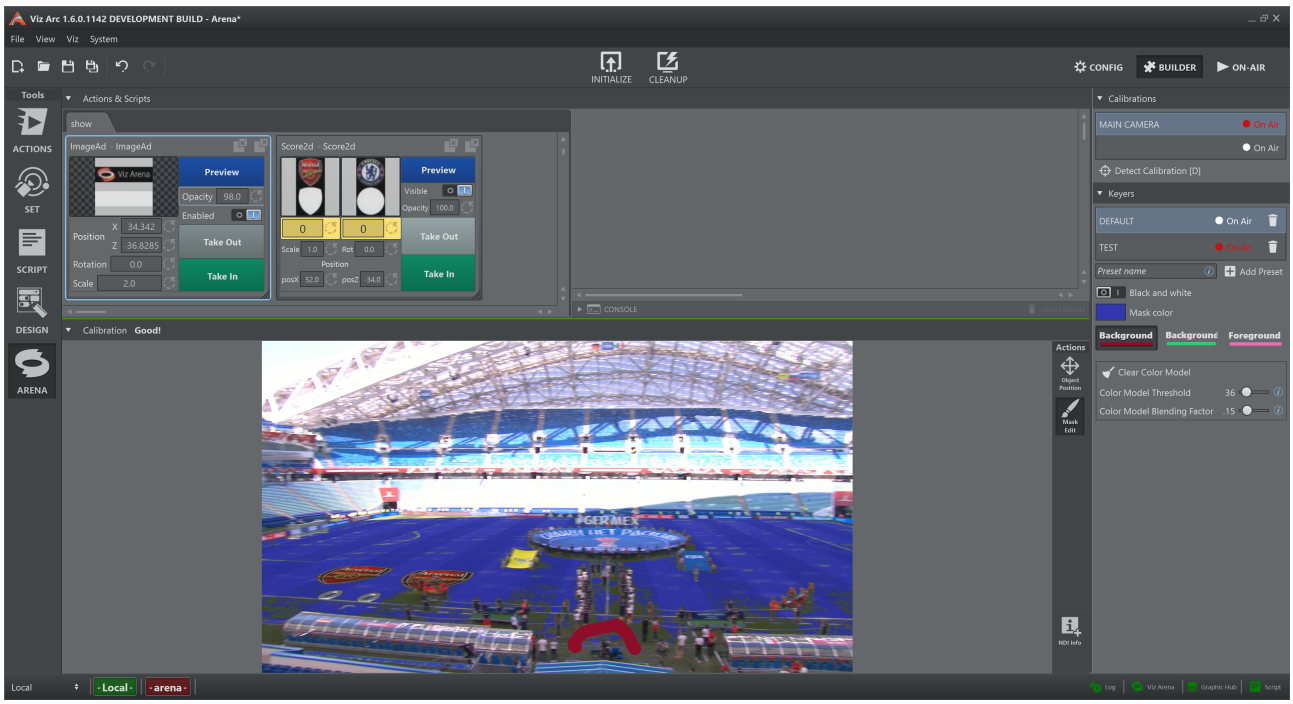
This action enables to position any selected control object (in the Action Panel) on the field (see image above). As a visual aid, the cursor turns into a cross when over the output view.

Copyright © 2025 Vizrt



Mask Edit

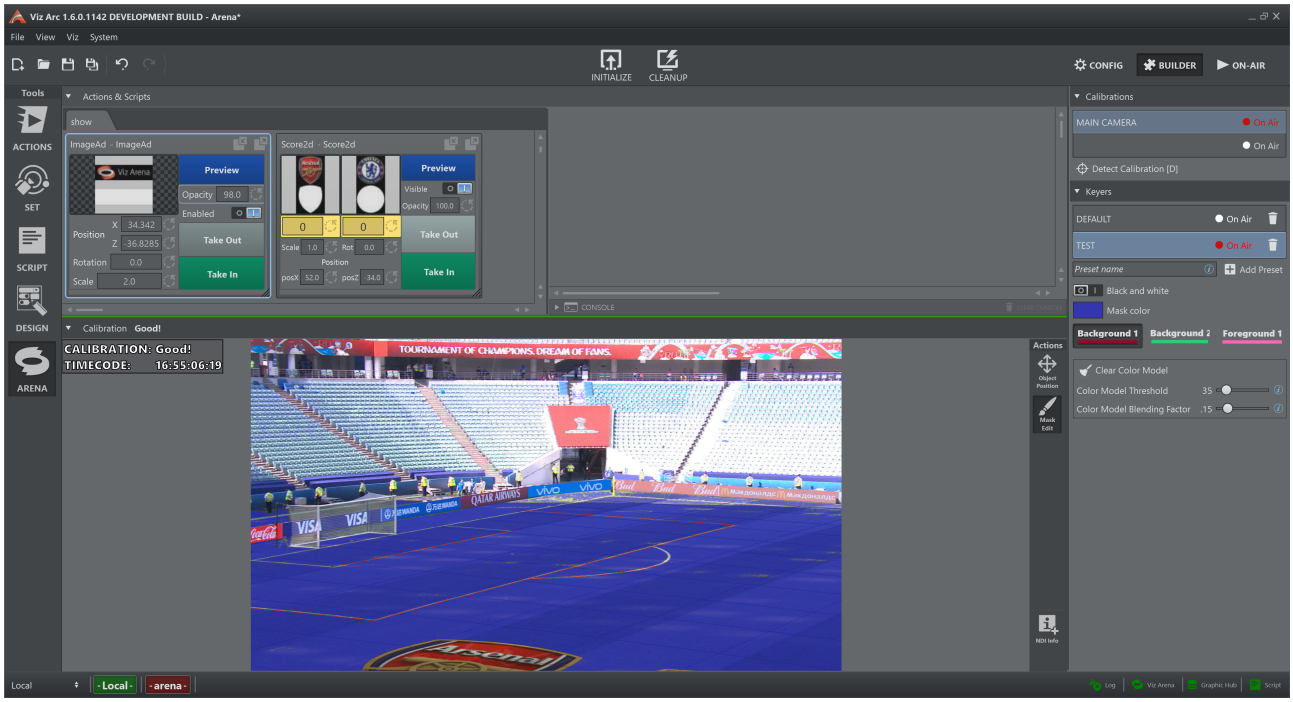
This action enables to adjust the color model used for calibration and by the **Viz Engine** for keying during a live game. Add brush strokes to improve the classification of the foreground and the background to refine the color model.

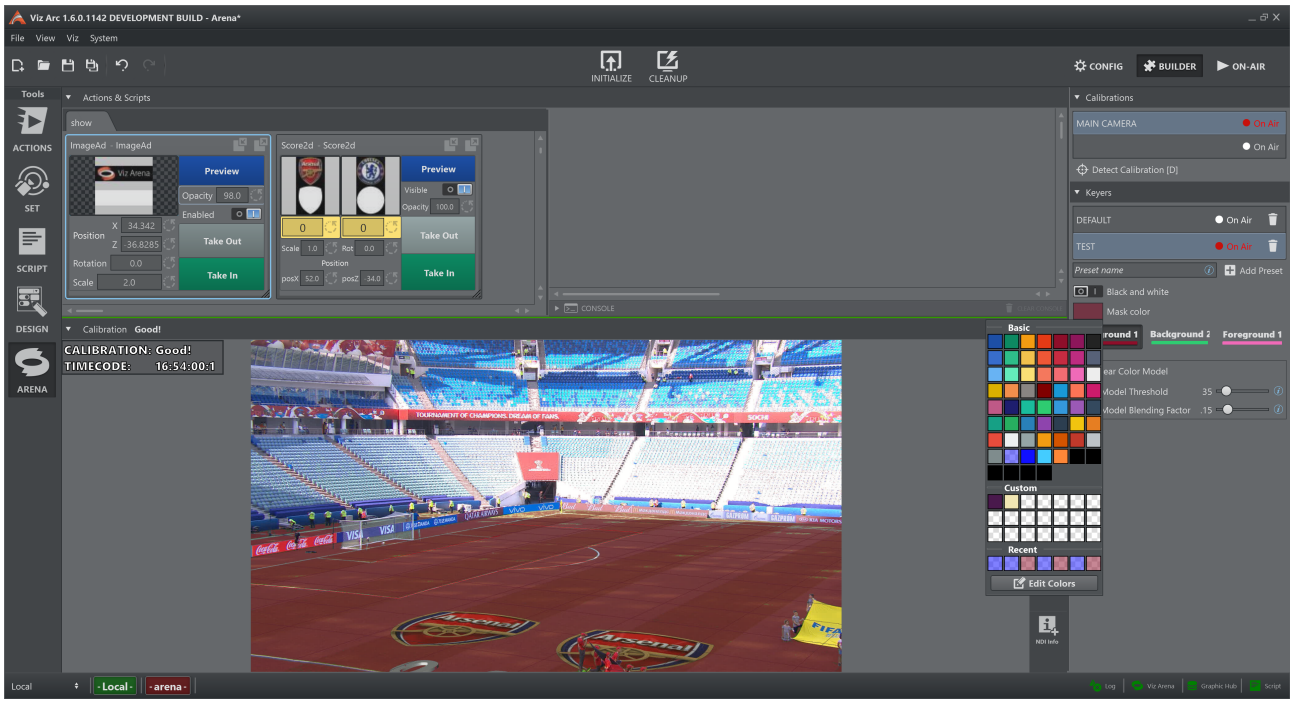


This action also turns on the visualization of the mask:

- Color mask

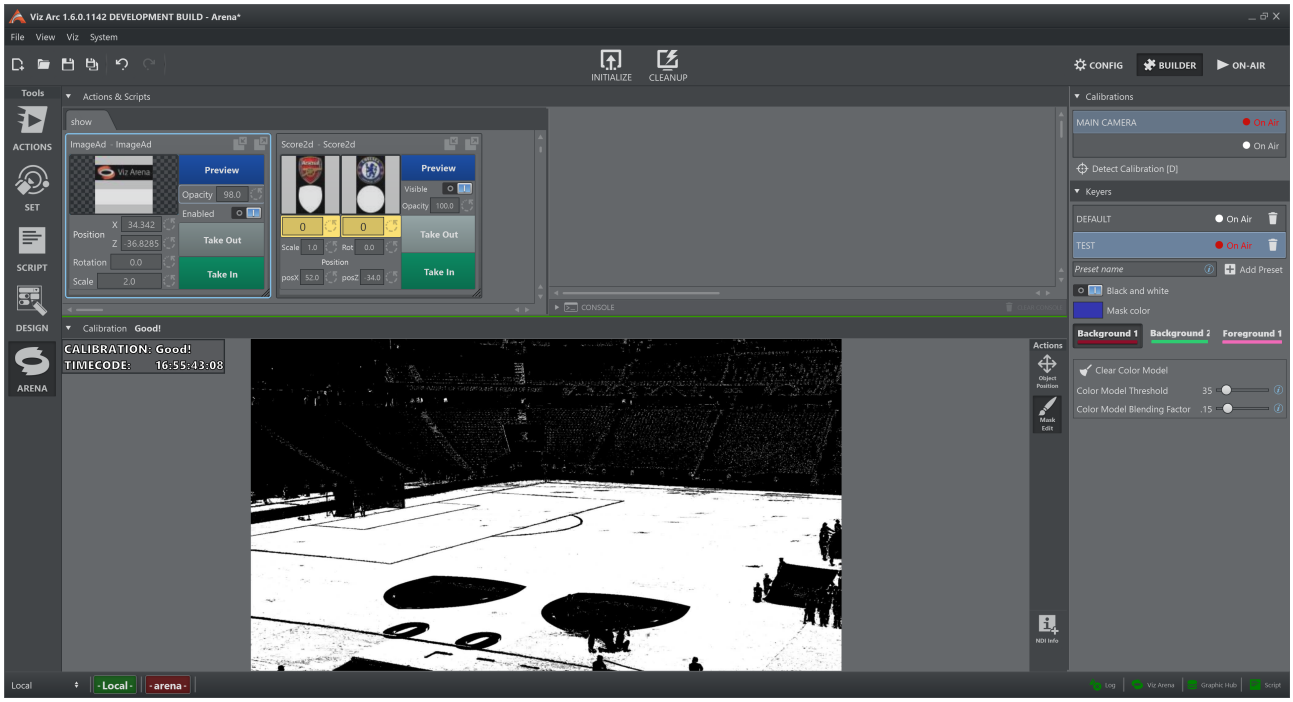
Copyright © 2025 Vizrt





- Black and white mask

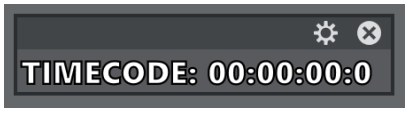
Copyright © 2025 Vizrt



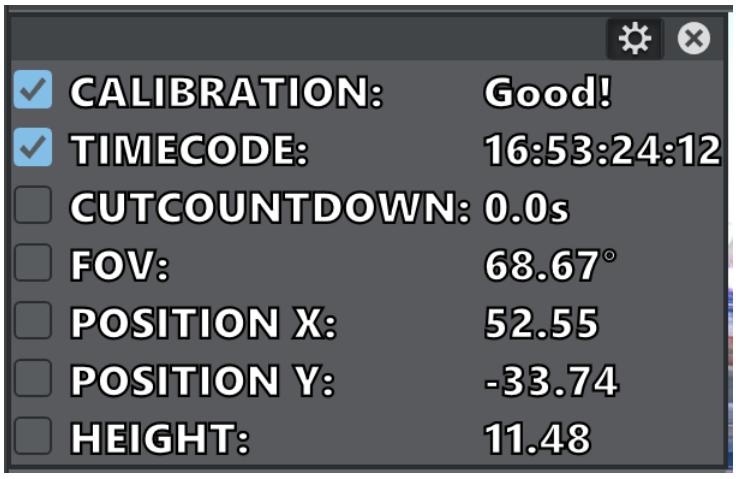
It is possible to toggle between the two modes see ***Settings panel > Mask > Back and White*** . During a live game, it is advised to regularly observe the output of the keying mask to ensure that the graphics are well rendered on Viz Engine and to provide the best possible tracking quality.

NDI Info

This button adds a new floating Info Panel to the interface.



- Click on the gear and the panel expands showing all the information that can be visualized:



Click again on the gear to exit from configuration mode.

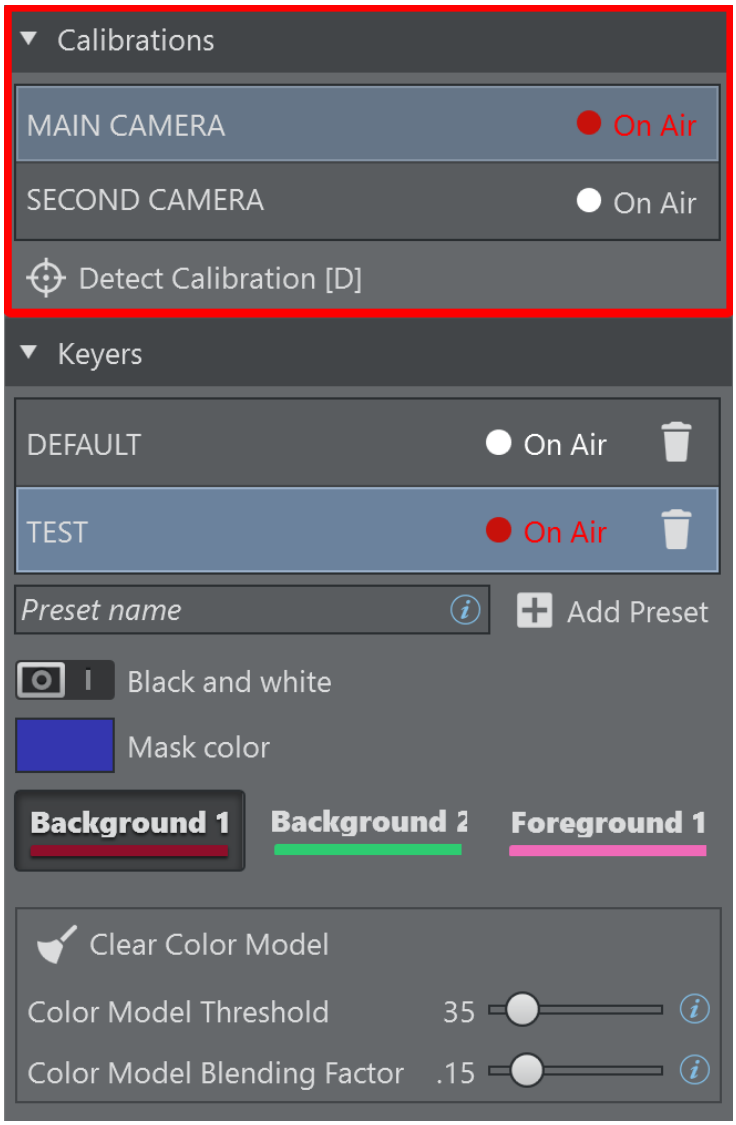
Copyright © 2025 Vizrt

- Click on the to close the panel.
- Grab the header to move the panel around and place it anywhere.

There is no limit to the number of panels that can be added.

Settings Panel

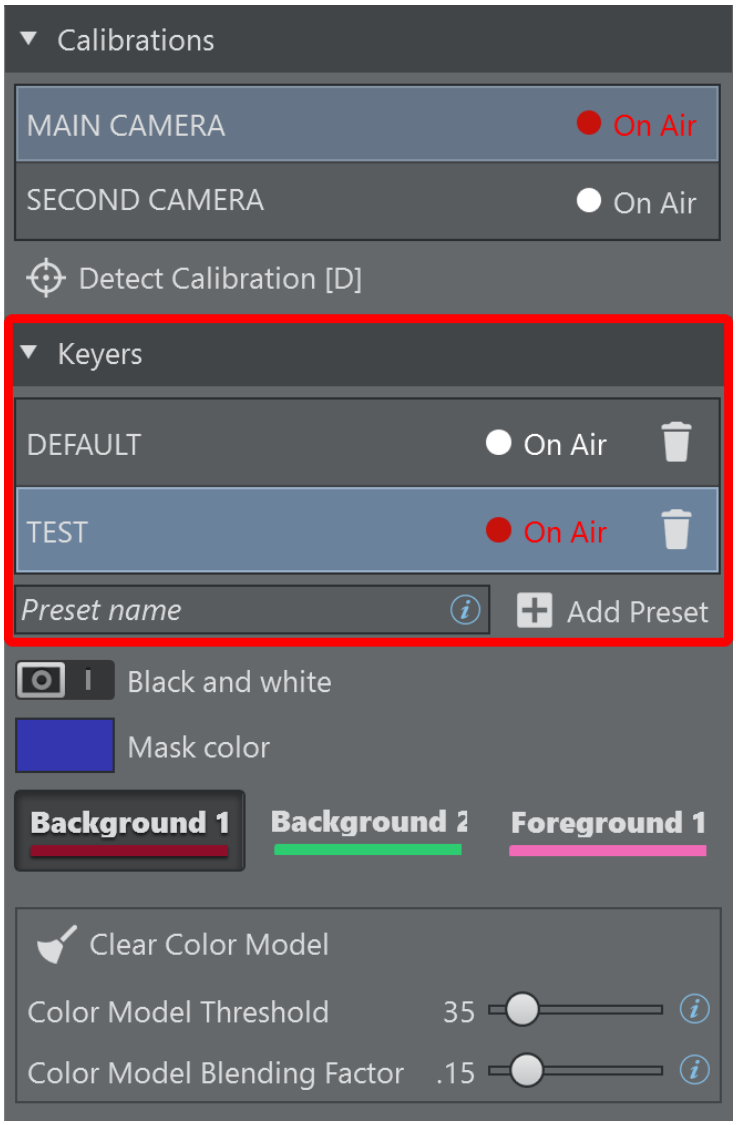
Calibrations



- Select a Camera/Calibration for both preview and program feed by clicking on the calibration name.
- **Detect Calibration [D]:** Detects the calibration of the current image. Click this button (or press the **D** button) if the tracked calibration is off from the actual field lines.
  The calibration can be cleared using the keyboard shortcut **BACKSPACE**.

Copyright © 2025 Vizrt

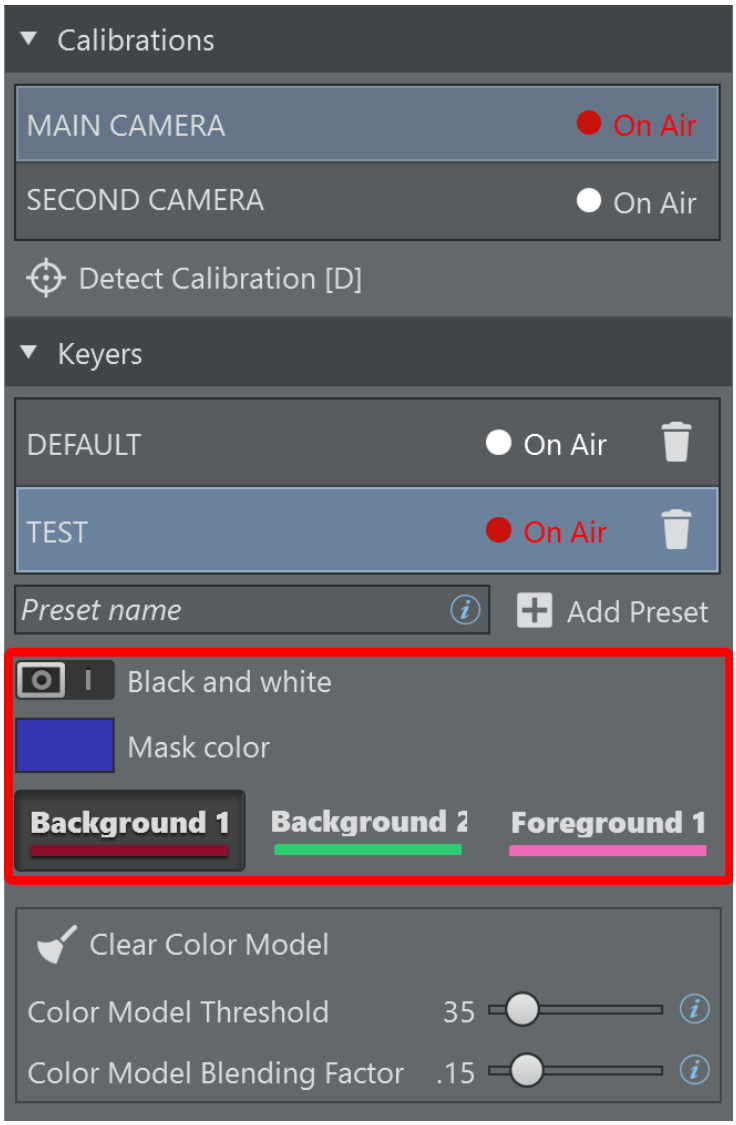
Keying Presets



- Select a preset for preview feed by clicking on the preset name.
- The preset that is currently On Air is marked with a red On Air label.
- If the On Air preset has any local changes that have not yet been applied to the output rendering, they are marked with an orange exclamation mark ⚠.
- Press On Air to update the output rendering.
- Press 🗑 to remove the selected preset from the list. If it is the preset On Air, the button does nothing.
- Press Add Preset to add a new empty keying preset to the list. The name is the one inserted in the text box next to the button.

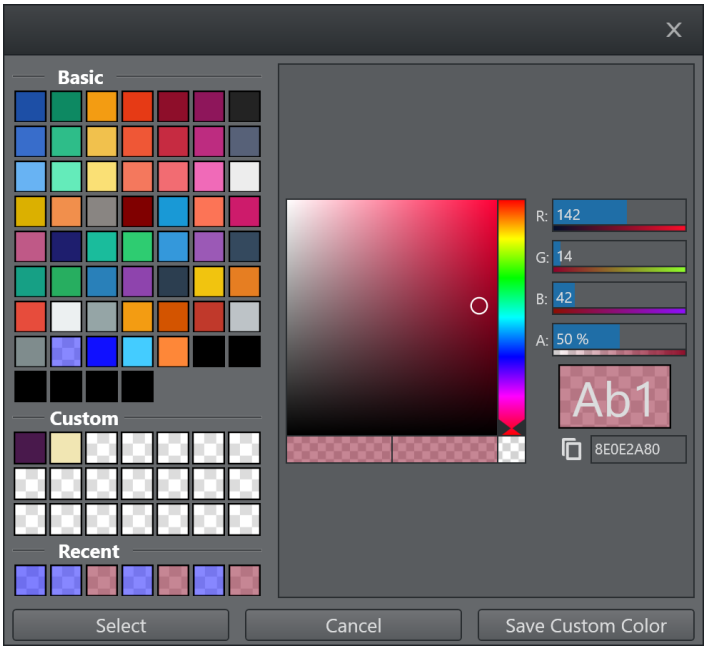
Copyright © 2025 Vizrt

Mask



- **Black and White:** Toggles the style of the keying mask.
- **Mask Color:** Selects the background keying mask color used for rendering which is most suitable for the playing field. Click on the colored rectangle and a new panel appears. Here it is possible to choose the mask color.

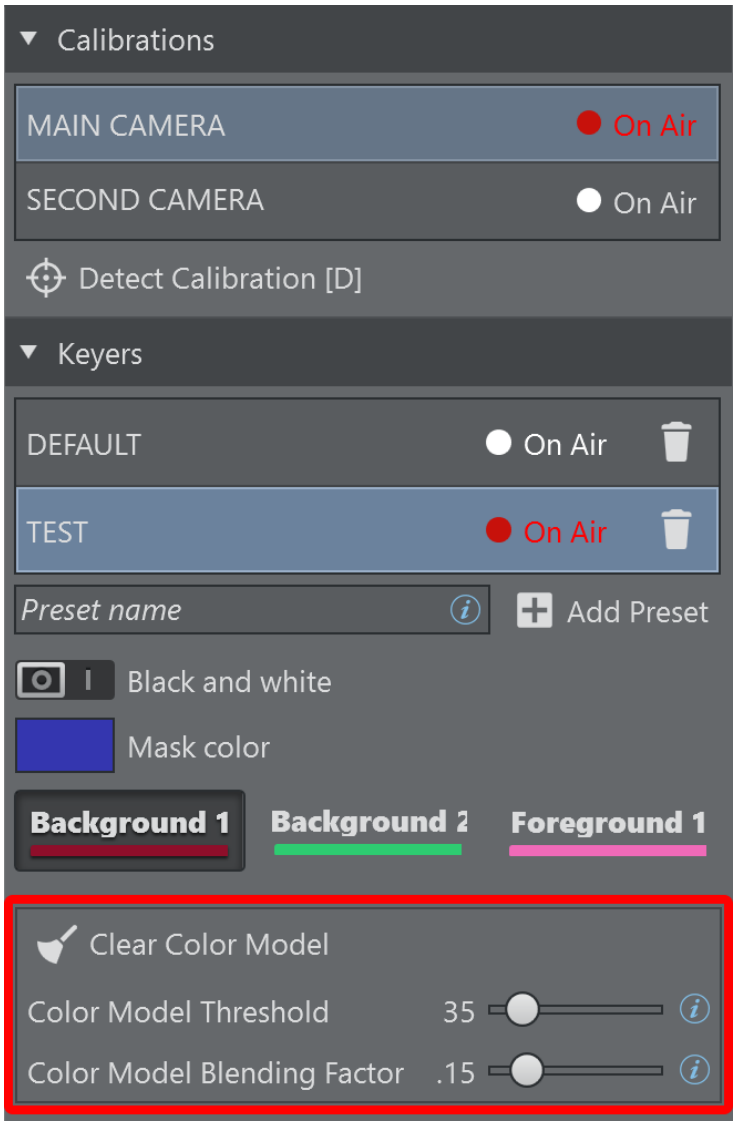
Copyright © 2025 Vizrt



- **Background 1:** Interacts with the background color model of the preset. Right click on the button and the color panel shows (see above), this changes the color of the stroke.
- **Background 2:** Interacts with the background logo color model of the preset. Right click on the button and the color panel shows (see above), this changes the color of the stroke.
- **Foreground 1:** Interacts with the foreground color model of the preset. Right click on the button and the color panel shows (see above), this changes the color of the stroke.

Copyright © 2025 Vizrt

Controls



- For a better understanding of each control please refer to the *Viz Arena User Guide*.
- Going over ⓘ with the mouse provides a brief description of the relative control.
- **Clear Color Model [C]:** Clears the stored keying information about the current preset and starts updating the color model from scratch. Alternatively, use the shortcut **C** on the keyboard.

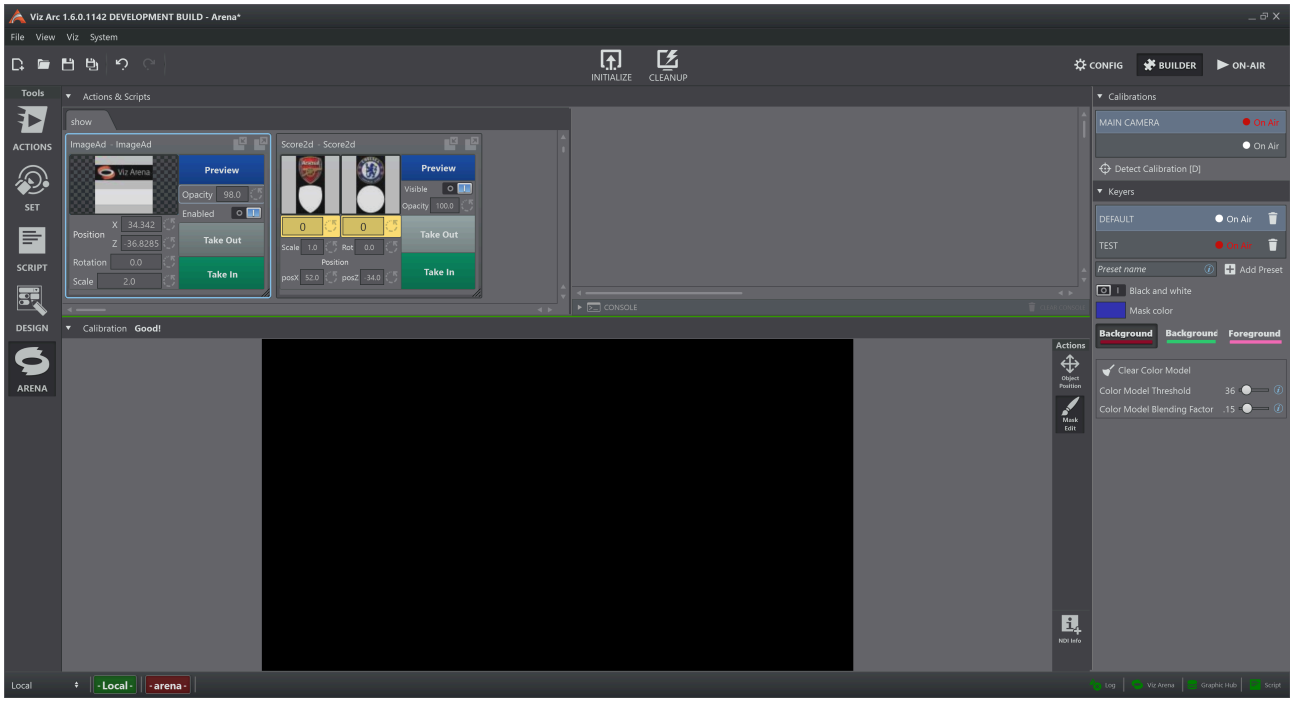
Startup

Using the command line parameter `--a` or `--arena`, it is possible to start Viz Arc directly with the Viz Arena tool visible.

Troubleshooting

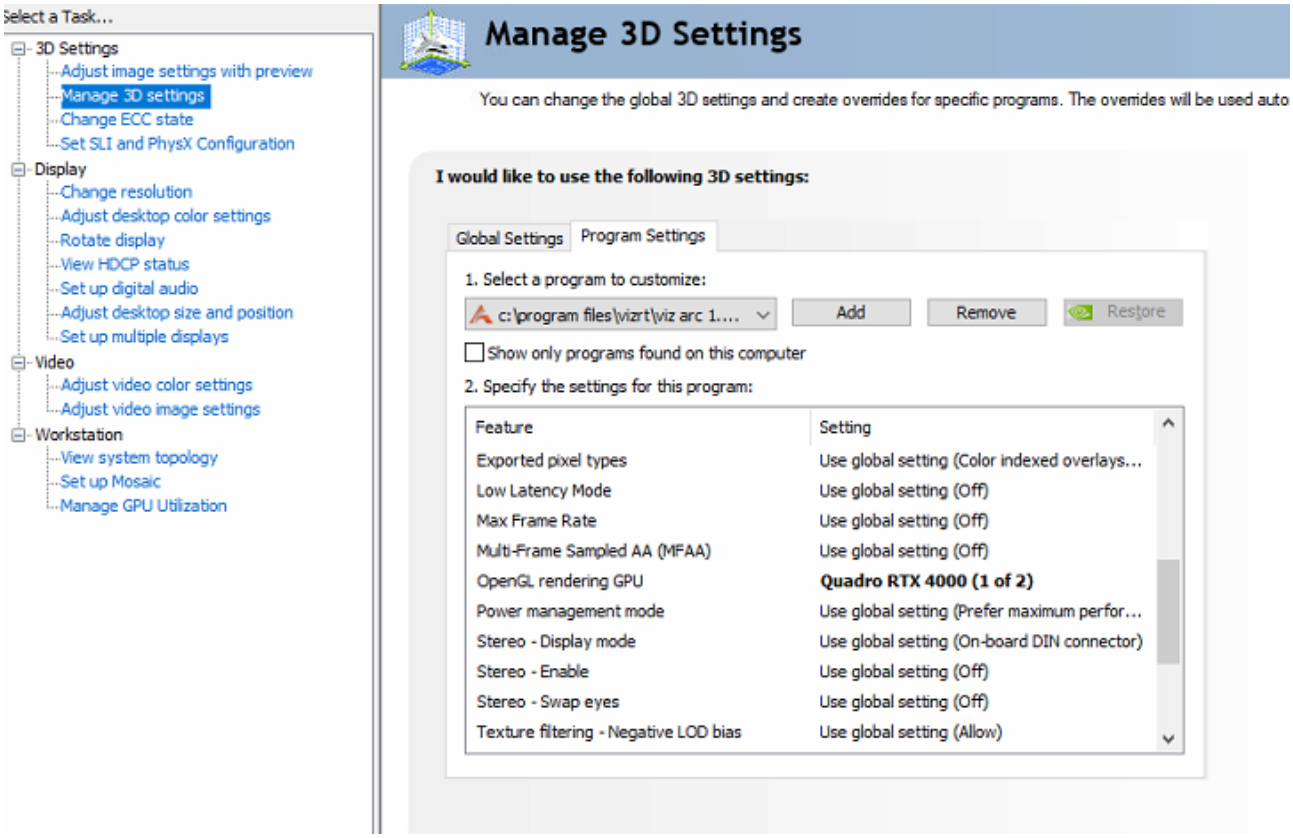
It may happen on multi-GPU machines that the output area remains black.

Copyright © 2025 Vizrt



In this case, configure the **OpenGL Rendering GPU** for the Viz Arc executable through the NVIDIA control panel.

Make sure to select the GPU that is connected to the connected Monitor. It might be assigned by default to the most powerful GPU in the system.

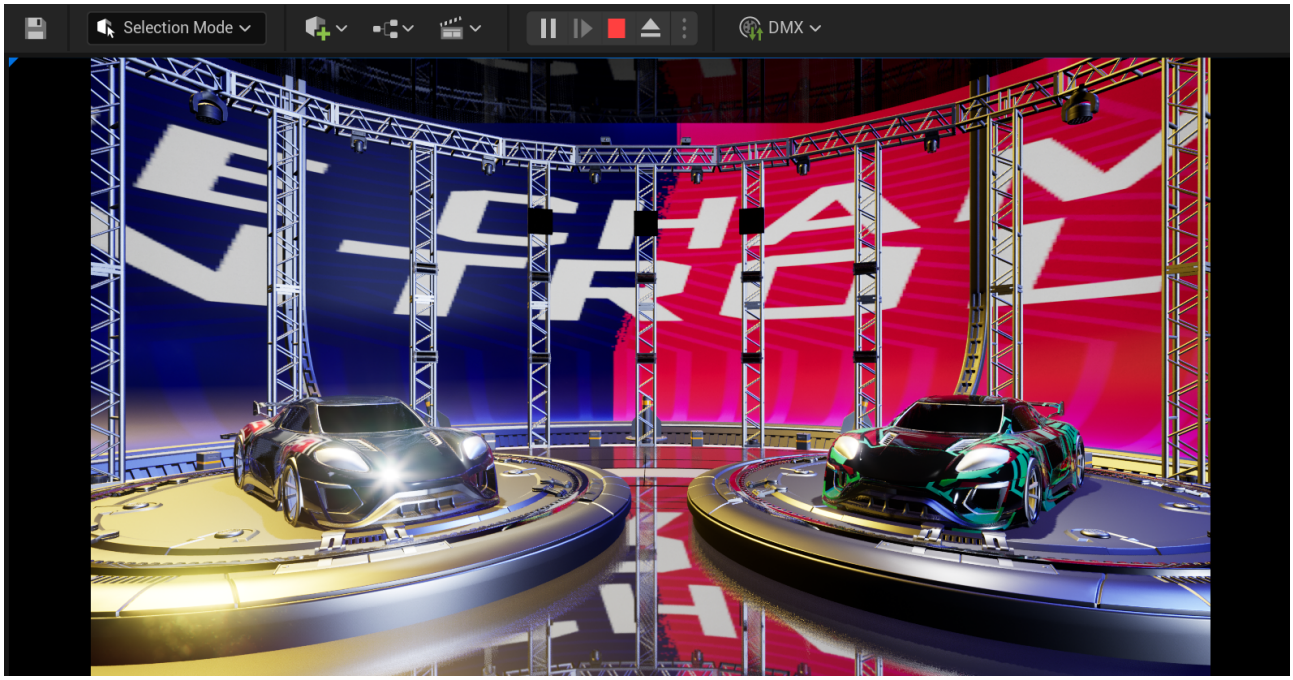


Copyright © 2025 Vizrt

5.8.6 Working with Unreal Engine

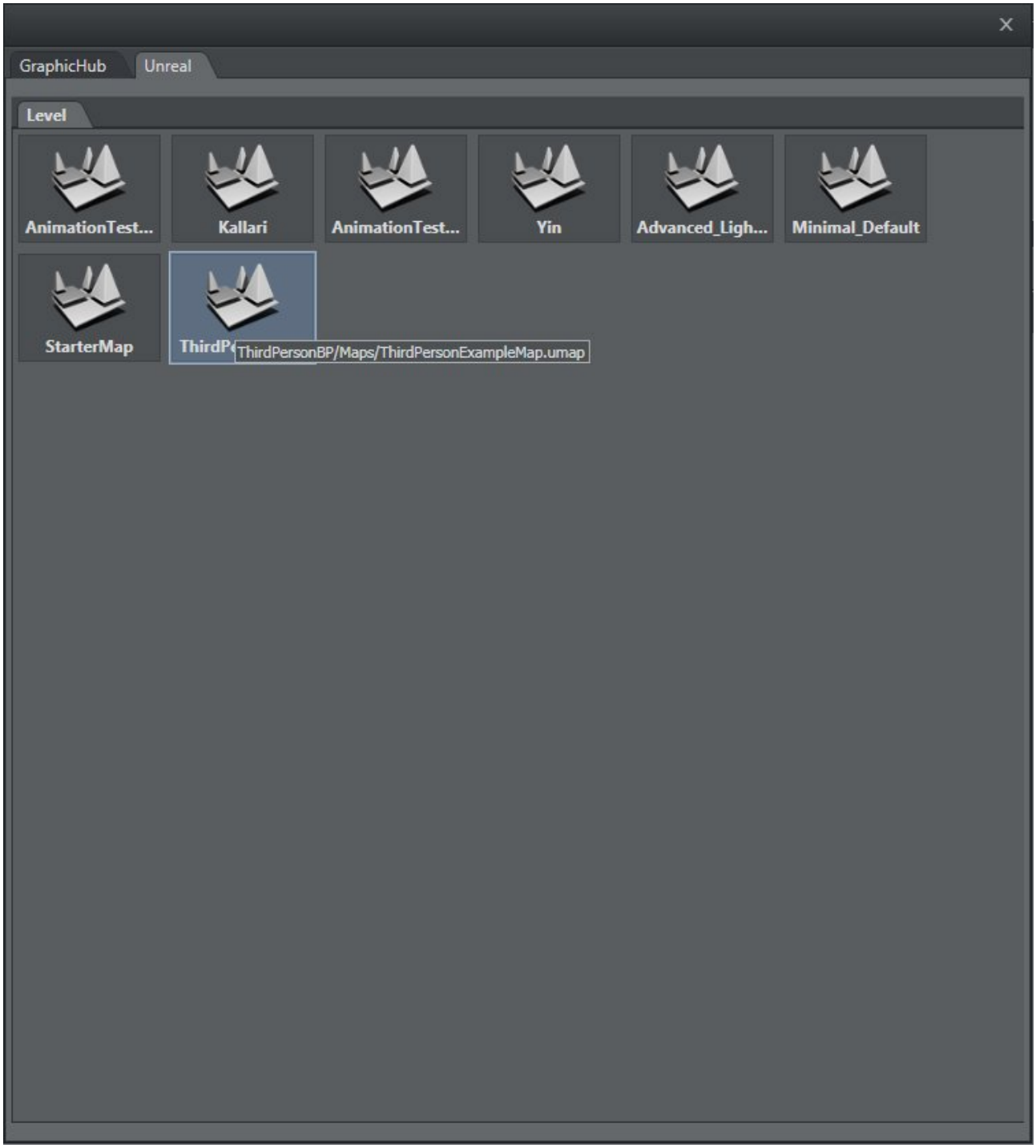
Viz Arc lets you import Unreal Levels from a specific project (selected and opened manually in Unreal Engine software).

> **Note:** Please see Configuring Profiles to correctly configure Unreal Engine in Viz Arc.

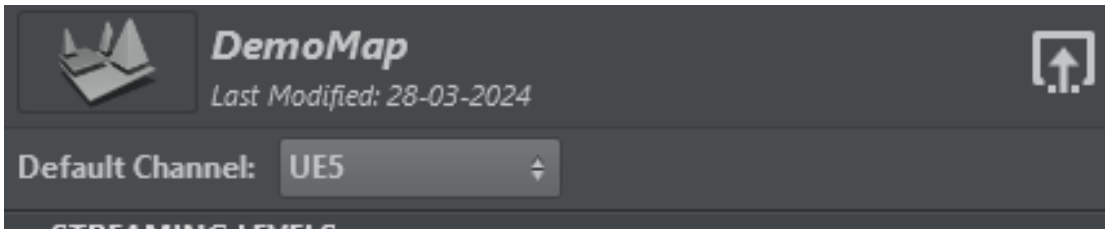


First, open the Unreal Engine software and select a project. Once the project is loaded, click the **Play** button to switch to Play mode.

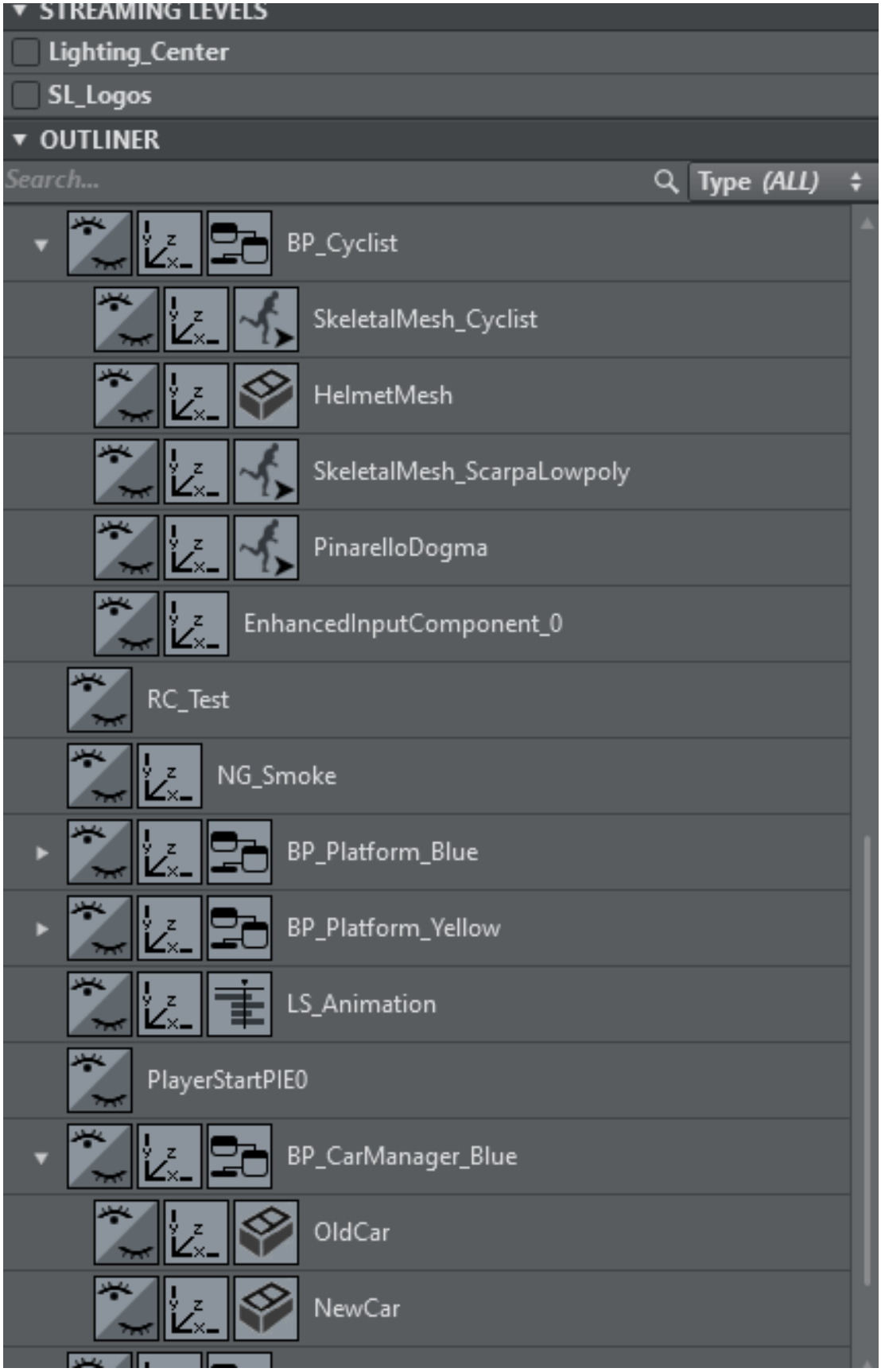
Copyright © 2025 Vizrt



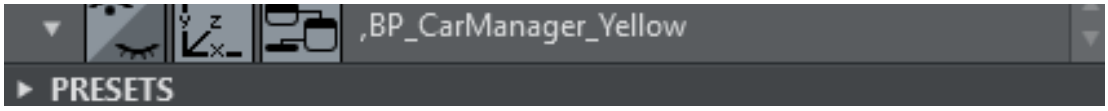
In Viz Arc, click the **ADD SCENE** icon on the left side of the window and select the Unreal panel.



Copyright © 2025 Vizrt

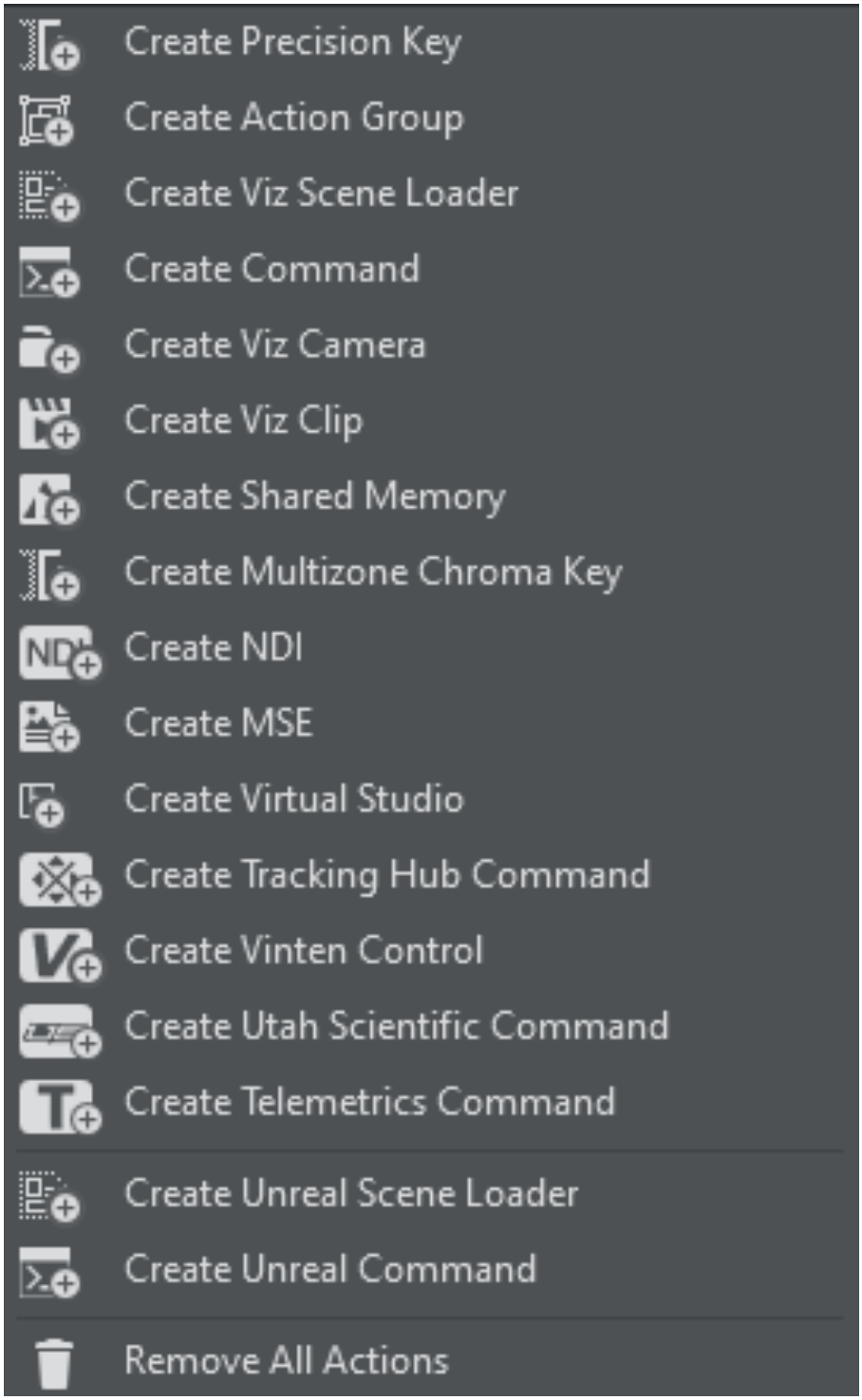


Copyright © 2025 Vizrt

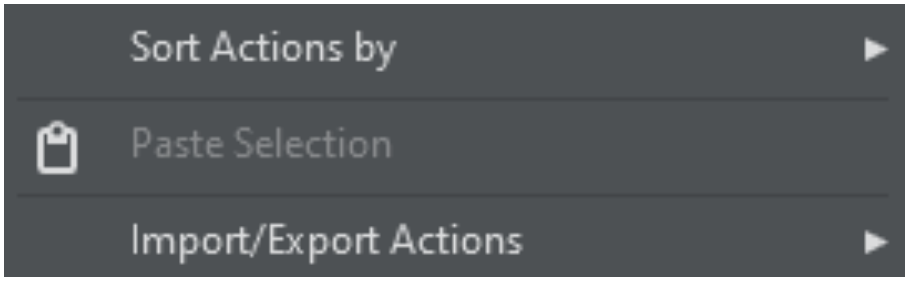


Once selected, you need to load the scene by clicking the **LOAD SCENE** icon in order to retrieve details.

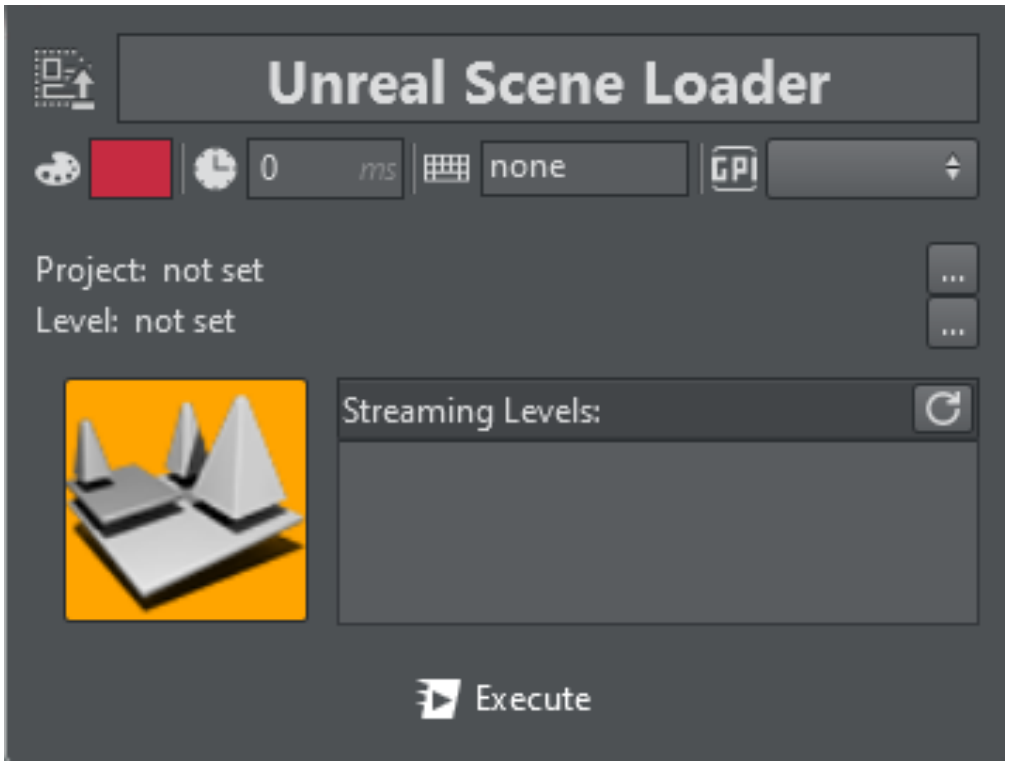
Double-click the **root** folder to show the list of elements. To create an action, drag and drop available content from the tree into the actions area.



Copyright © 2025 Vizrt

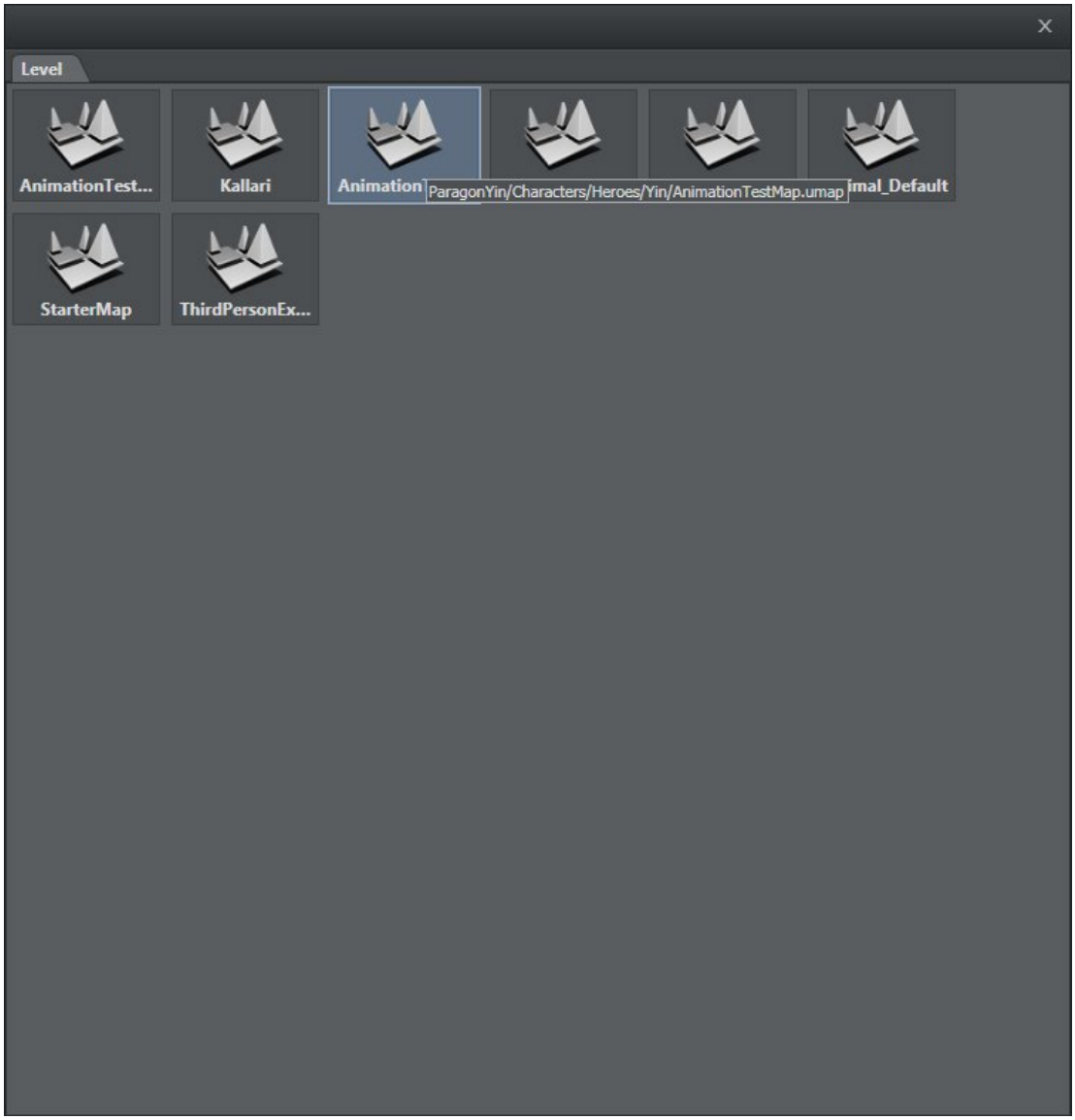


Load an Unreal scene by right-clicking on an empty space in the actions area.



Once the relative action has been created, in edit mode, click **Select Unreal Level** to open the browse window.

Copyright © 2025 Vizrt



Select the desired level and double-click it to load.

See Also

- Supported Unreal Actions

Copyright © 2025 Vizrt

Supported Unreal Actions

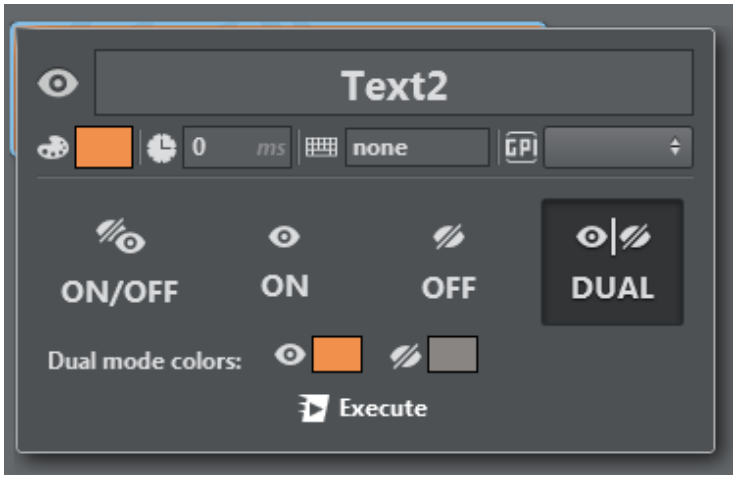
Viz Arc recognizes the following types of Unreal actions:

- Scene Loader
- Active
- Position/Rotation/Scaling
- Sequence
- Animation
- Text
- Mesh
- Light
- Presets
- Blueprint
- Level Blueprint
- Dispatcher

Scene Loader

Read more about the Viz Arc Engine Service.

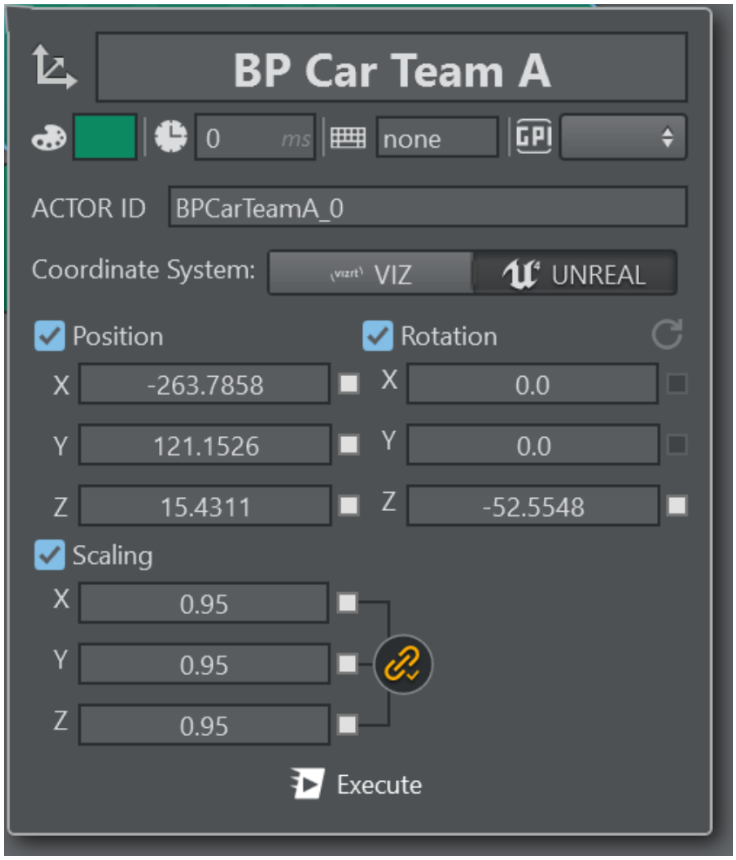
Active



Active action lets you switch a container to visible or invisible. It's possible to set the container to **ON**, **OFF** or **TOGGLE** between the two.

Copyright © 2025 Vizrt

Position/Rotation/Scaling



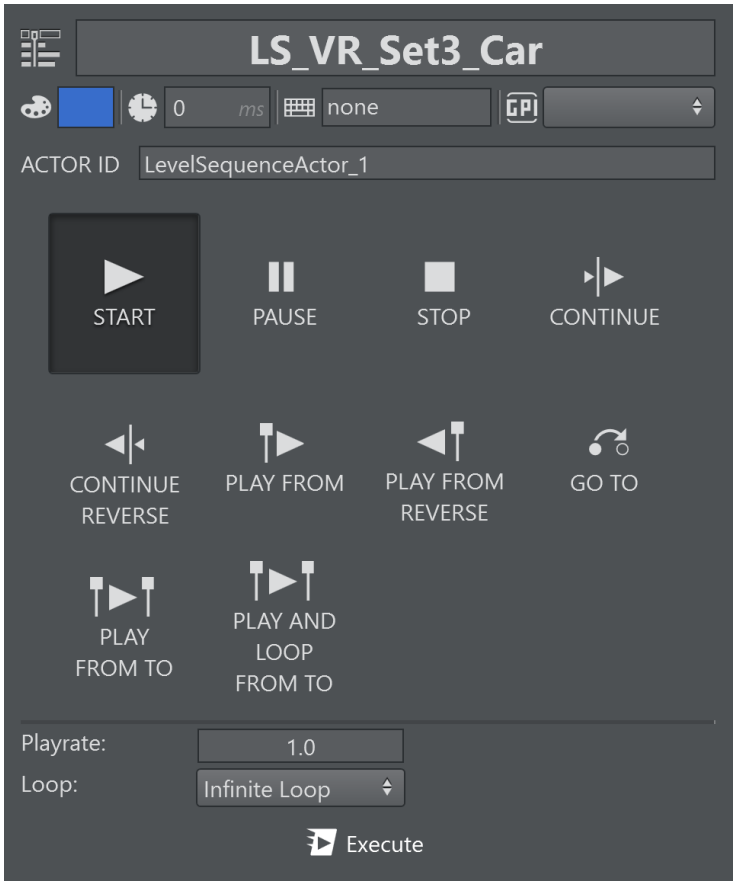
This action lets you set the position, rotation and scaling of the selected container. It's also possible to scale proportionately by clicking the **lock** icon  next to the scaling panel.

It is also possible to change Position and Rotation of a Streaming Level.

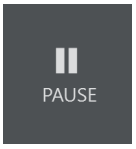
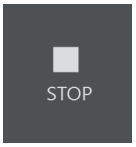
The coordinate system toggles the behavior of the positional XYZ axis, the default **UNREAL** is a Z-up left-hand system, while **VIZ** is Y-up right-hand system. It might come handy to use the Vizrt coordinate system when objects needs to be positioned relative to each other in the two engines.

Copyright © 2025 Vizrt

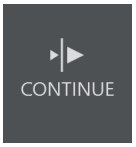
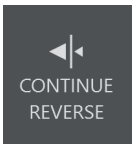
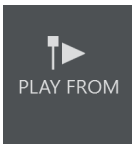
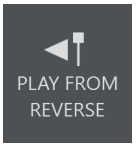
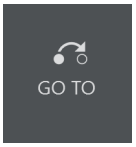
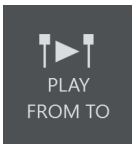
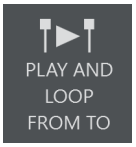
Sequence



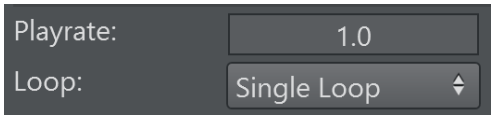
A Sequence action lets you control a sequence level created in Unreal Editor:

	**START** animation.
	**PAUSE** an animated director.
	**STOP** an animated director.

Copyright © 2025 Vizrt

	**CONTINUE** an animation to the next step.
	**CONTINUE** in **REVERSE** mode.
	**PLAY from** a specific KEYFRAME.
	**PLAY from** a specific keyframe in **REVERSE** mode.
	**GO** directly to a specific KEYFRAME.
	**PLAY** from a specific keyframe to a specific keyframe.
	**PLAY** from a specific keyframe and then loop between two specific keyframes.

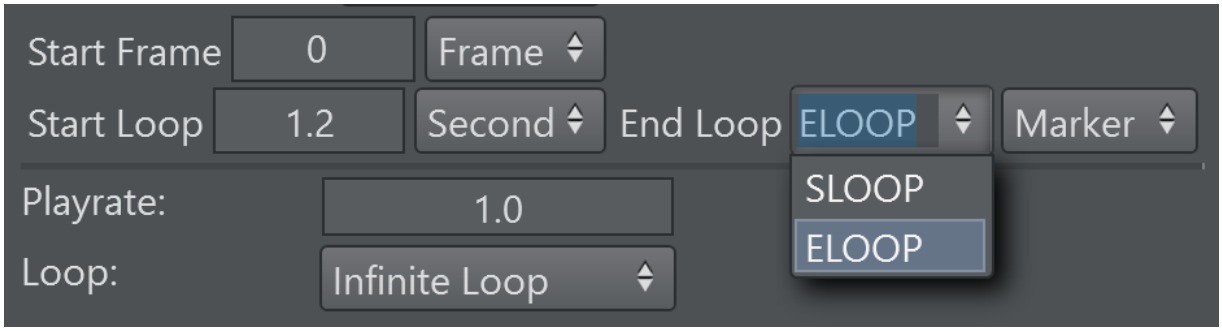
The play-rate and the number of loops can be set on any play action selected.



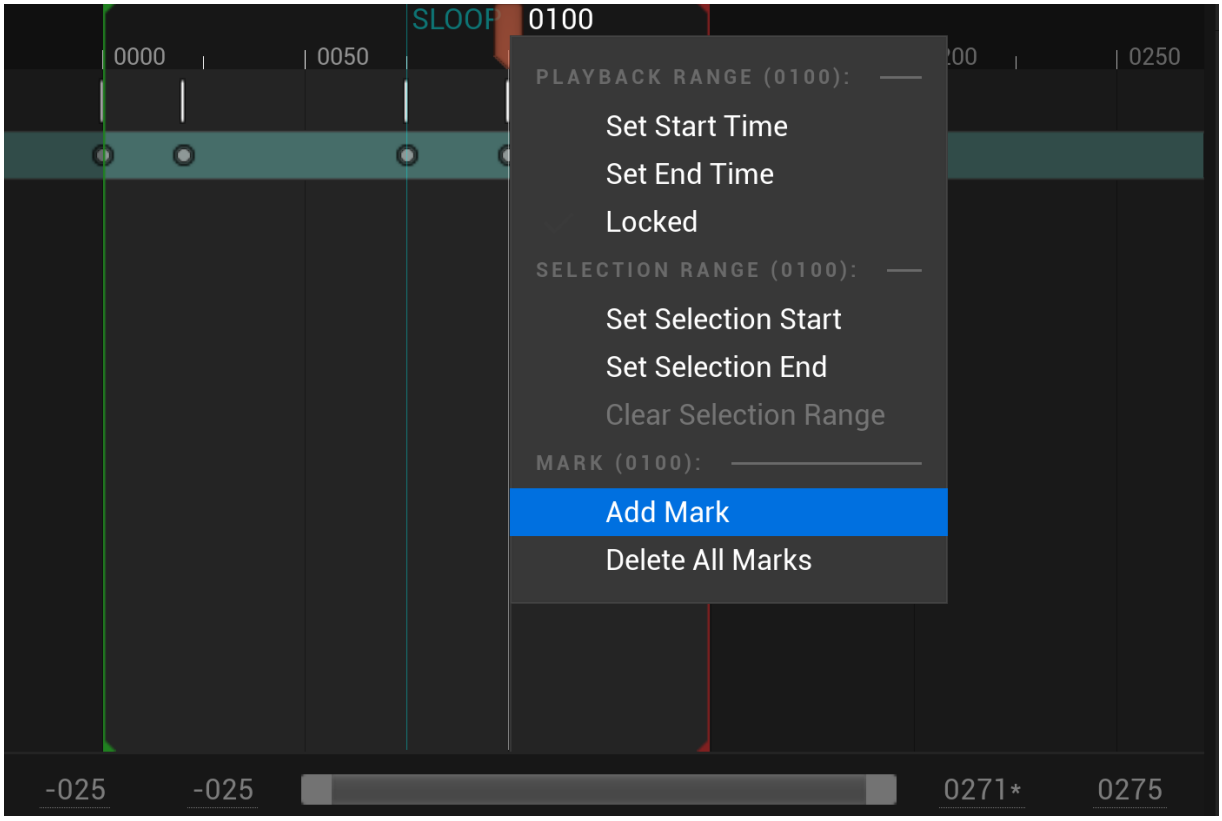
A play-rate of `1.0` represents normal playback rate while a play-rate of `2.0` means double playback speed.
Select as loop mode either **Single Loop**, **Multiple Loops** (specifying the number of loops) or **Infinite Loop**.

Clicking **PLAY FROM**, **PLAY FROM REVERSE**, **GO TO**, **PLAY FROM TO** or **PLAY AND LOOP FROM TO** makes a menu appear where you can set the position in **frames**, **seconds** or as a **marker**.

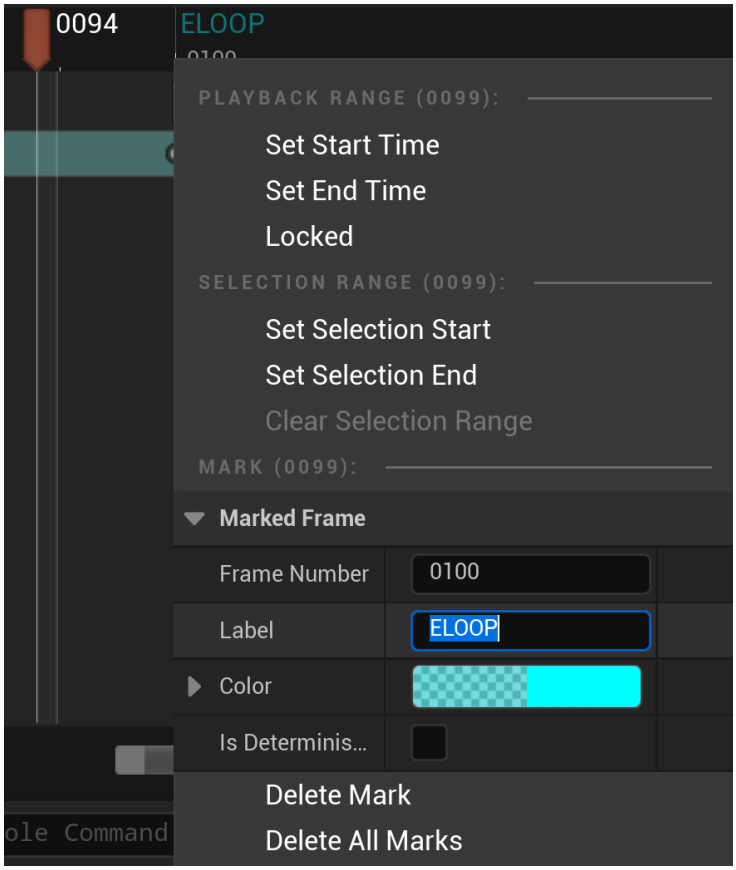
Copyright © 2025 Vizrt



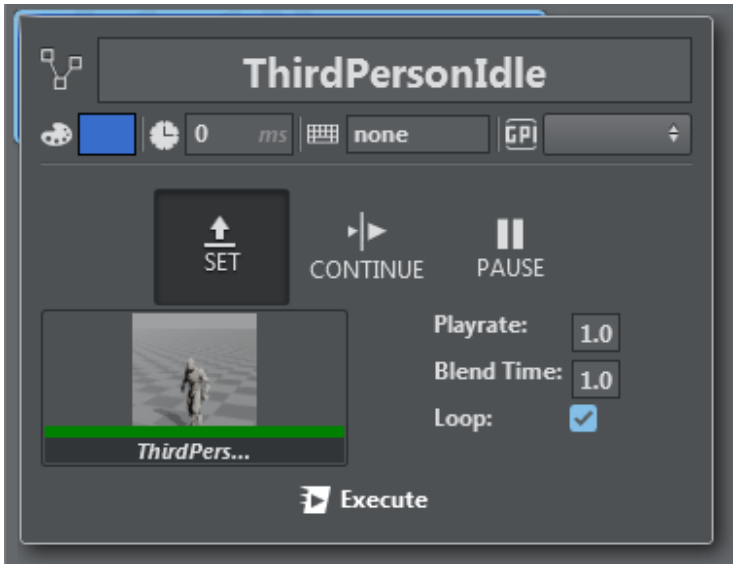
A **Mark** can be set on the unreal sequencer editor by right clicking the timeline and selecting **Add Mark** or in case a marker already exists, its label and position can be modified.



Copyright © 2025 Vizrt



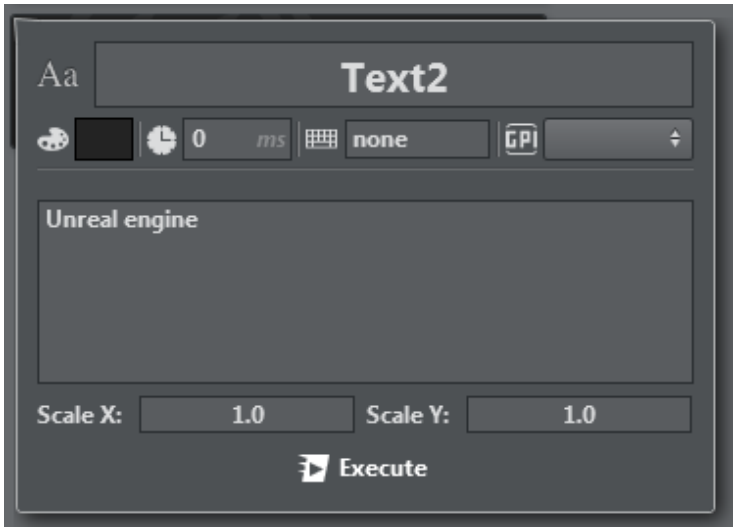
Animation



Animation action lets you assign an animation to the selected Unreal actor or browse a custom animation from those available by clicking the **Animation** icon. Animations can also be played, continued, stopped and/or looped.

Copyright © 2025 Vizrt

Text

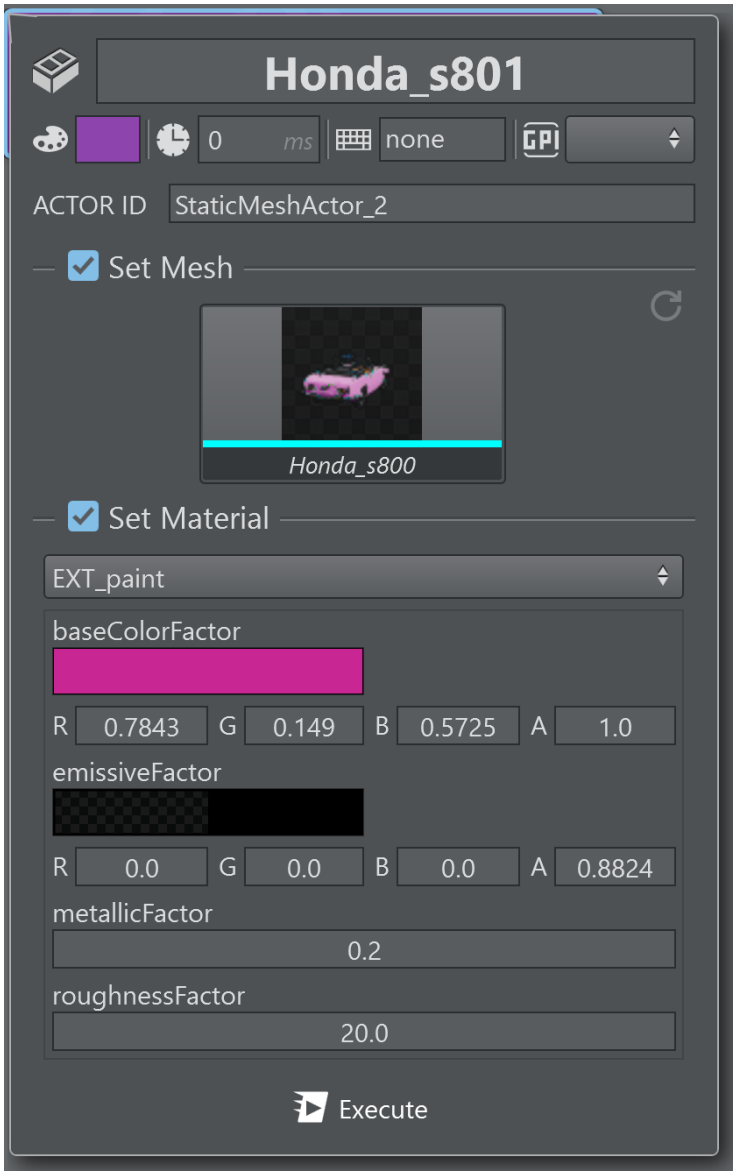


Text action lets you modify a text element in an Unreal level and set X and Y scaling.

Mesh

This action lets you change the static mesh and the material or material instance properties of a **Static Mesh Actor**.

Copyright © 2025 Vizrt



Use the checkbox **Set Mesh** when the action should set the static mesh and check the **Set Material** when the action should set any specific material or material instance property.

Mobility Property on Unreal Actor: 

The mobility attribute on the unreal actor needs to be either **Movable** or **Stationary** to make the mesh editable.

Light

This action lets you change lights propertis such as Intensity, Temperature and more specific ones.

Copyright © 2025 Vizrt



There are four different types of light action:

- Directional Light
- Spot Light
- Point Light
- Rect Light

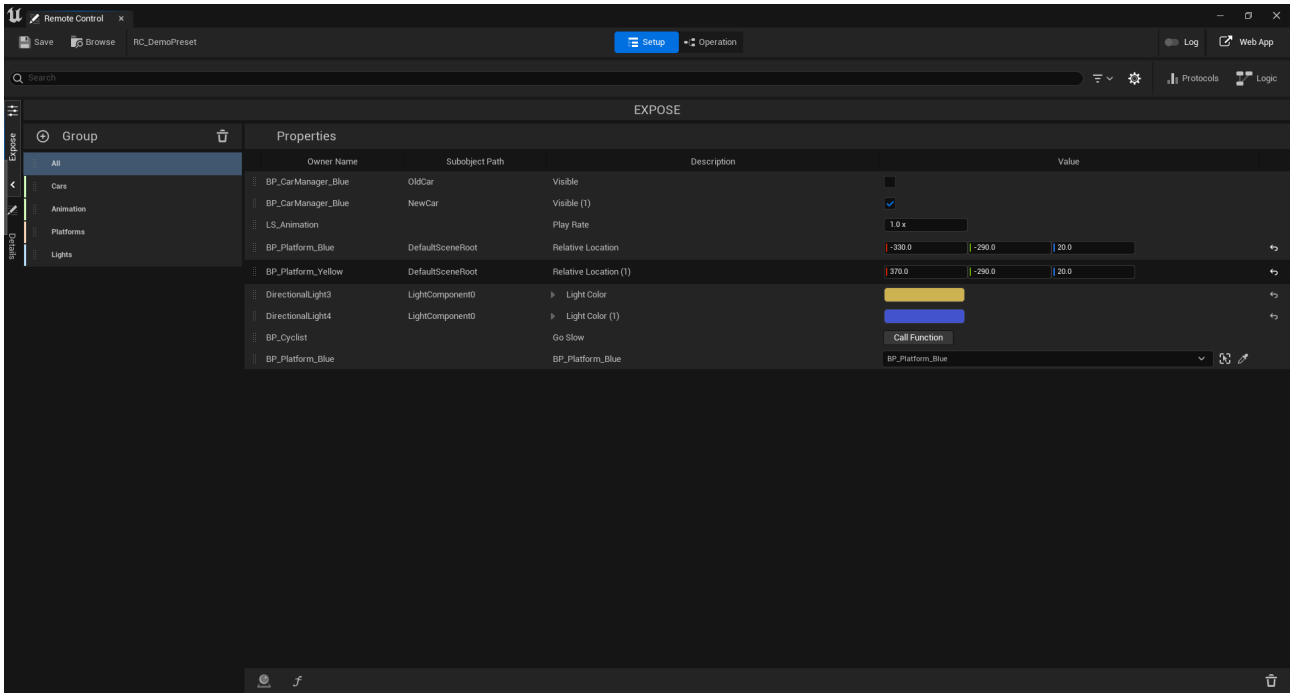
Arc automatically recognizes them and creates an action with specific porperties.

Presets

Unreal Engine Version 4.26 introduces **Remote Control Presets**. This allows the user to tag easily attributes on the outliner to be exposed to external control applications. Those presets can be set using a **Preset Action**.

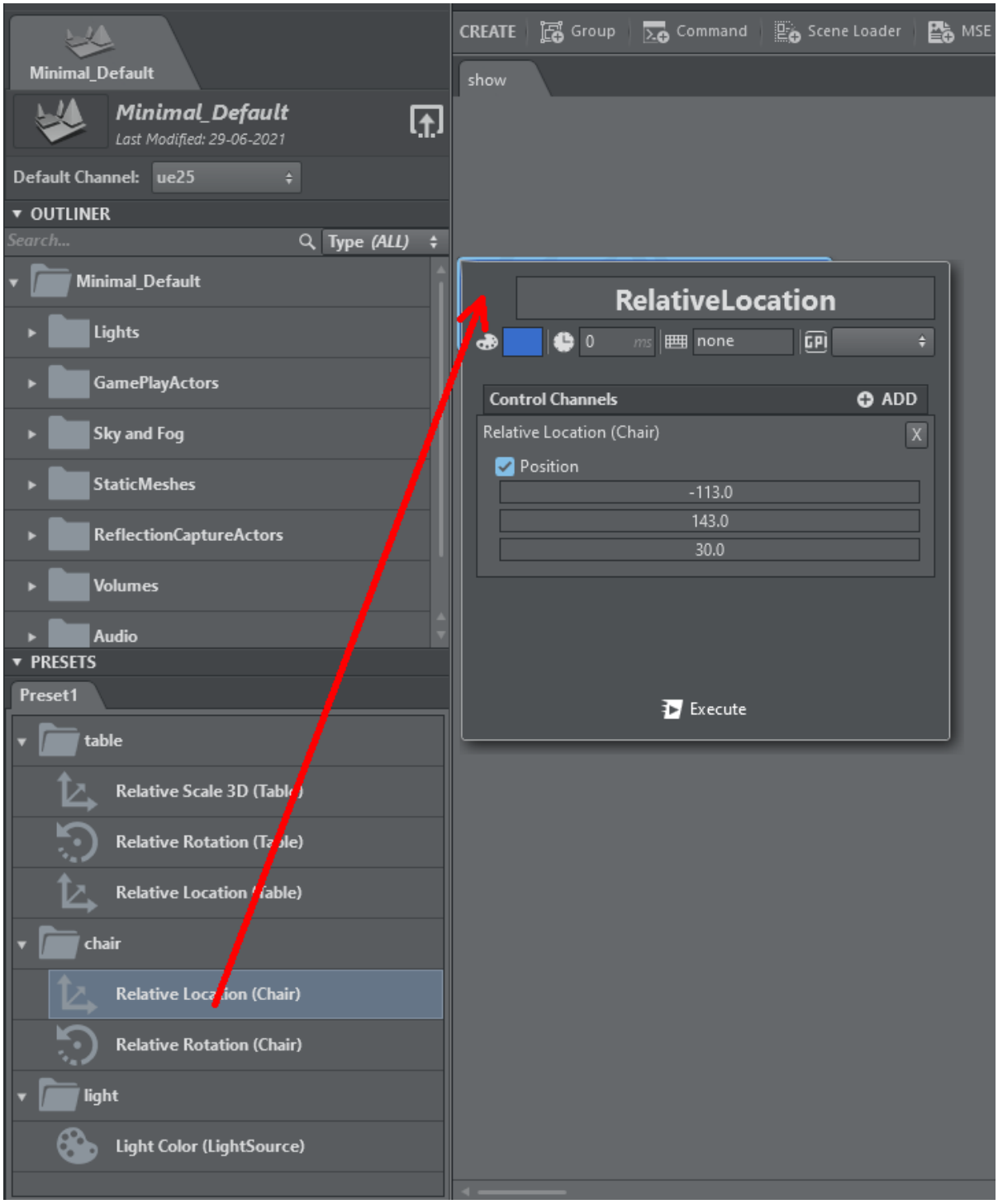
A preset on the Unreal Engine might look like this, exposing transformations of object, light color, a blueprint function or visibility of a mesh.

Copyright © 2025 Vizrt



When loading a level into Viz Arc, additionally to the outliner, the presets are displayed below in the **PRESETS** panel.

Copyright © 2025 Vizrt



Drag any of the presets to the action canvas to create a **Preset Action**.

Make sure that `-RCWebControlEnable -RCWebInterfaceEnable` are added to the `UELoaderCommandLine` registry variable of the Unreal launcher.

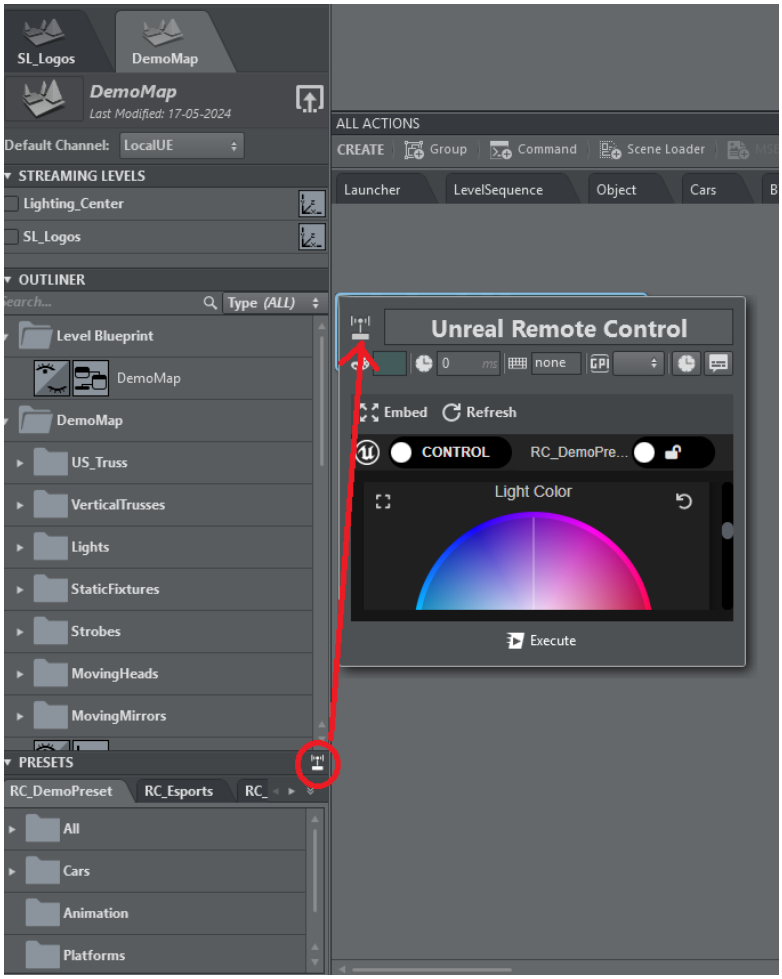
Copyright © 2025 Vizrt

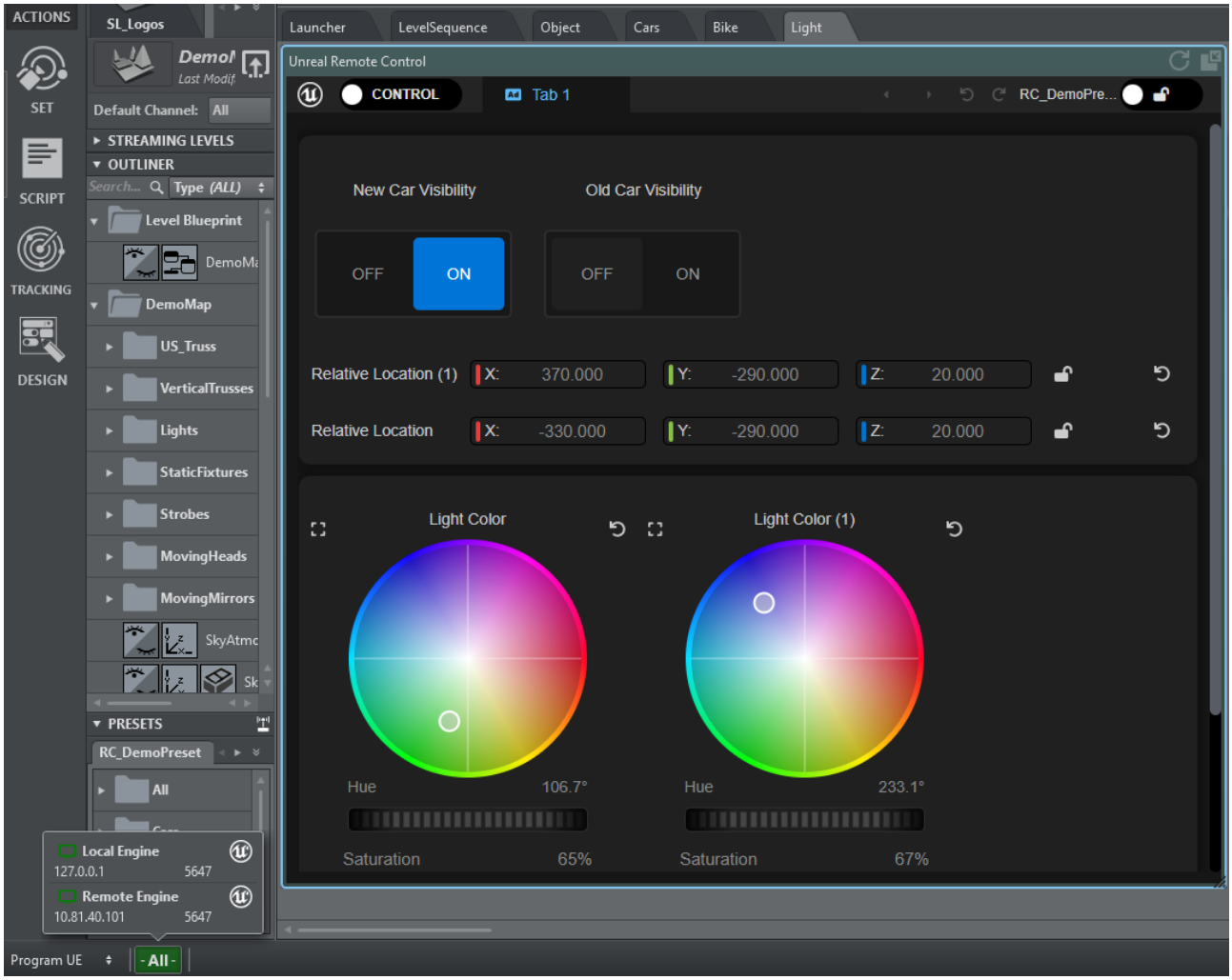
Unreal Engine's Binding Address: By default, the Unreal Engine's web interface binds to the local network interface only, such that only *127.0.0.1* or localhost addresses work with the **Preset Action**. To start the web service on all available network interfaces add this section to the *DefaultEngine.ini* of your Unreal project:

```
[HTTPServer.Listeners]
DefaultBindAddress = 0.0.0.0
```

For more information, see the https://dev.epicgames.com/documentation/en-us/unreal-engine/remote-control-for-unreal-engine.

If Unreal Web Interface is enabled, we can use it as action.





Drag the icon to the action canvas to create a **Remote Control Action.**

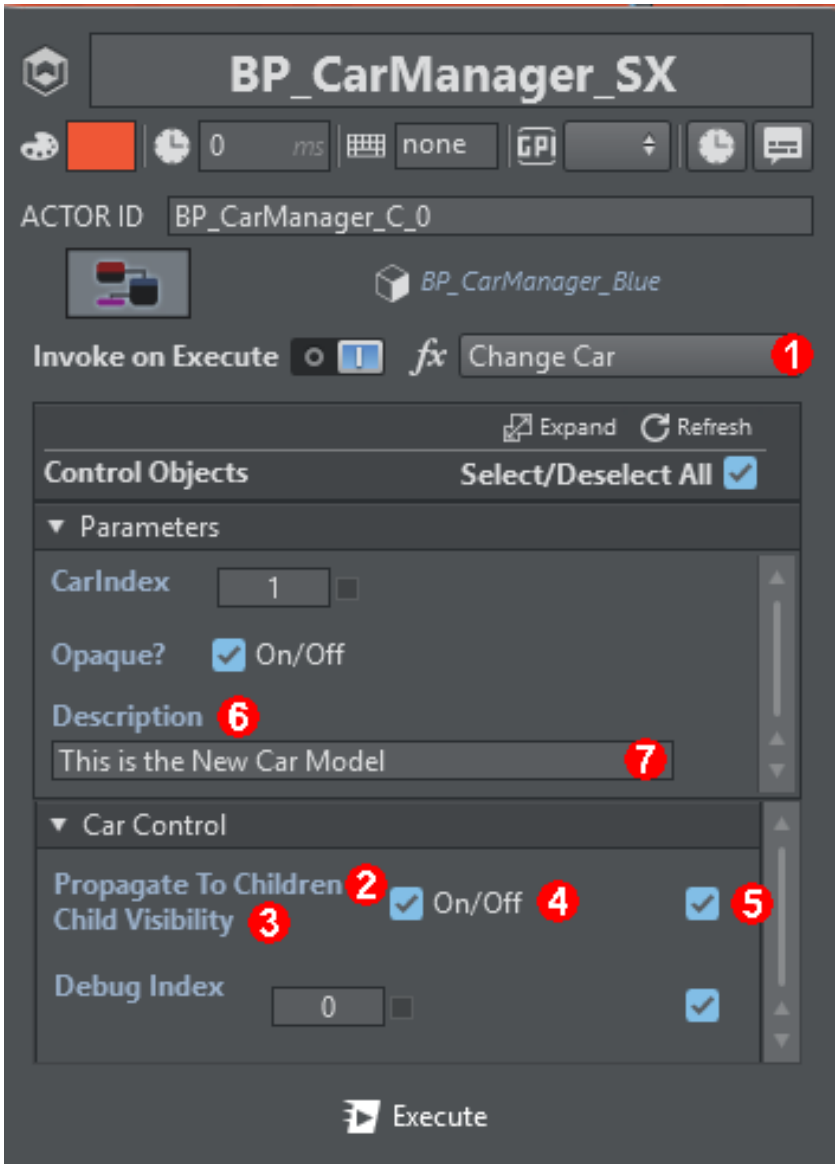
Any changes made in the view is sent to all Unreal Engines in the selected channel that have an active web interface. It is very important that both scenes have the same preset with the same properties.

If you lose connection to a Unreal Engine web interface or want to establish a new one, just click on the **Refresh** button.

Blueprint

A Blueprint action can be used to set public Blueprint variables and invoke functions on it.

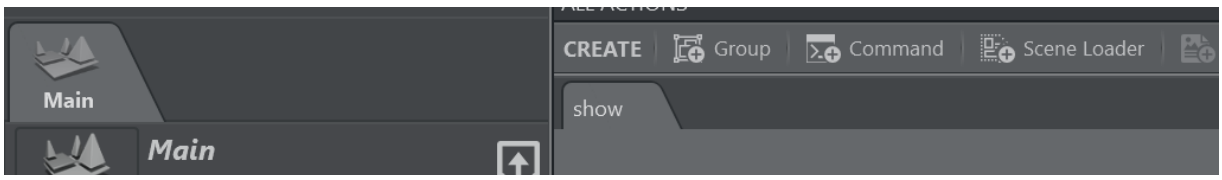
Copyright © 2025 Vizrt



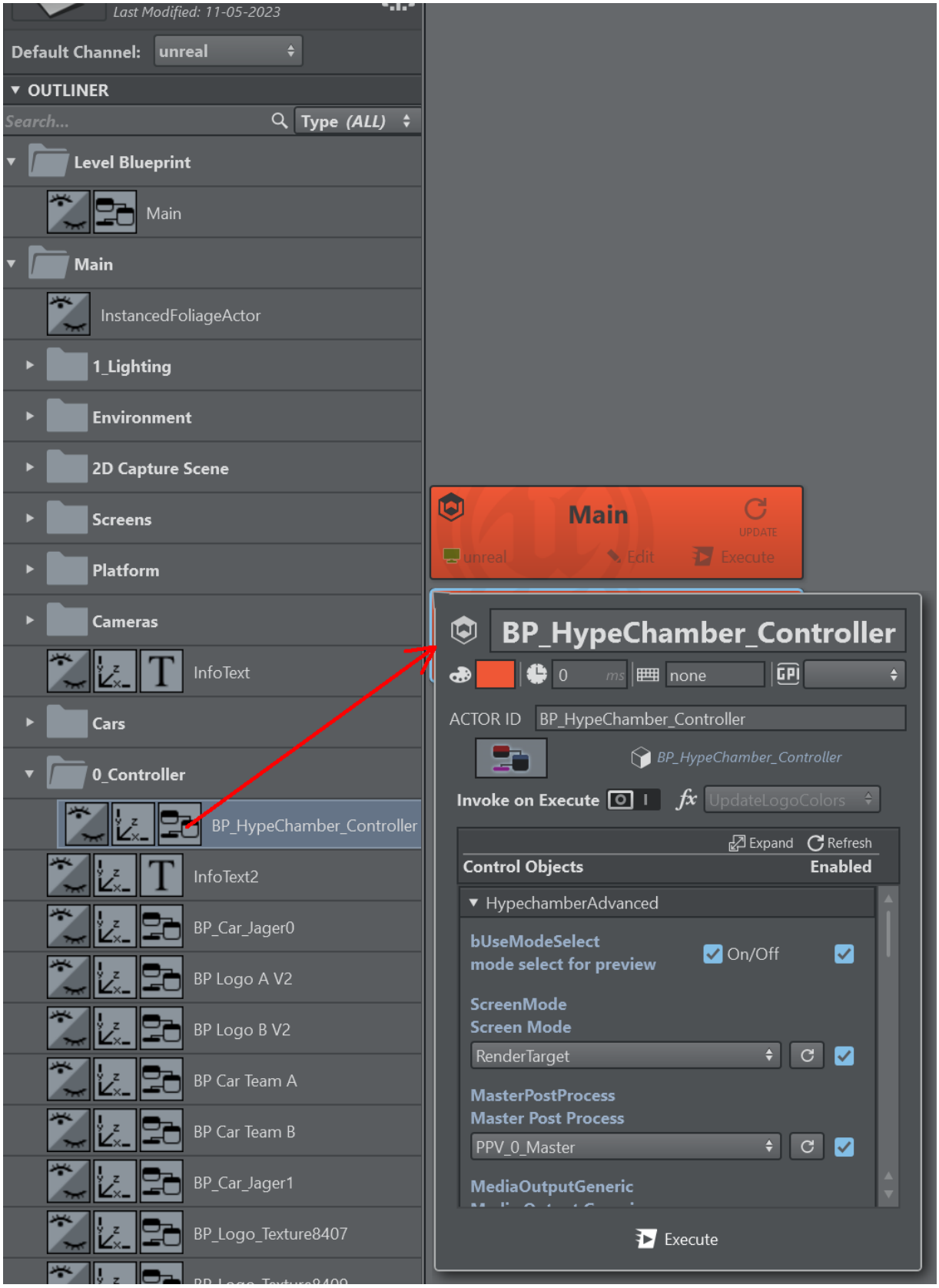
1. If enabled, the function to be called when executing this action.
2. The variable name.
3. The Tooltip of the variable.
4. The actual value.
5. Enable or disable setting this variable.
6. Function parameter name (if function has parameters).
7. Function parameter value to send (if function has parameters).

A **Blueprint Action** can be created by dragging and dropping the blueprint icon from the outliner to the action canvas. The **Level Blueprint** can be found in the outliner view

under the **Level Blueprint** folder.

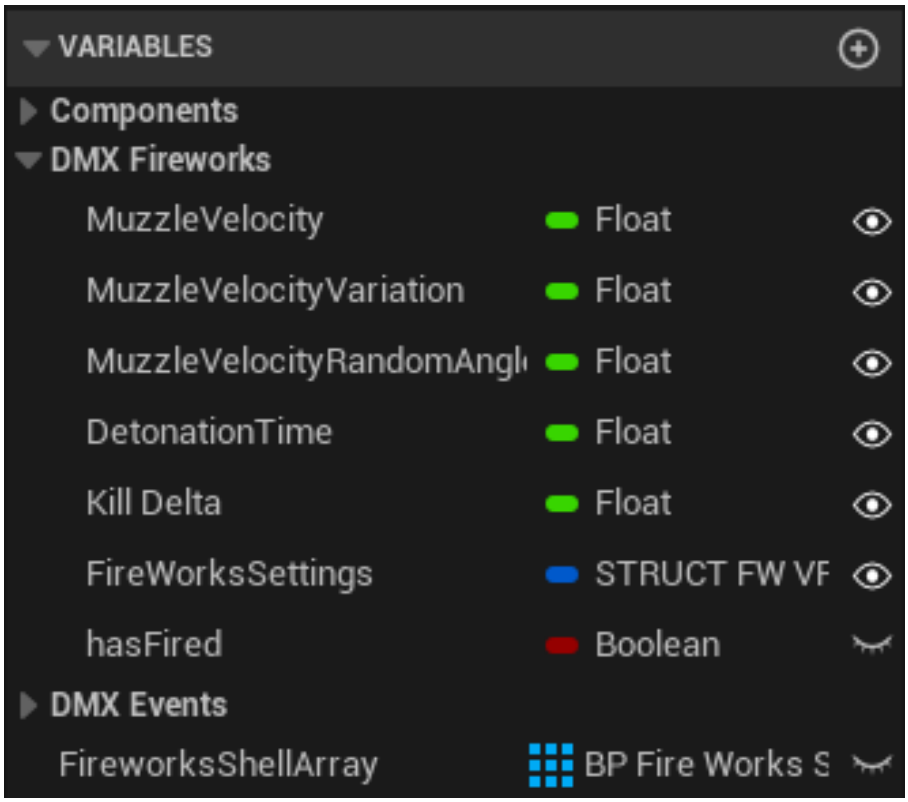


Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

Blueprint action contains a list of visible values (set as **public** and **editable** in Unreal). To make these values visible and accessible to Viz Arc, you must activate them in the Blueprint window:

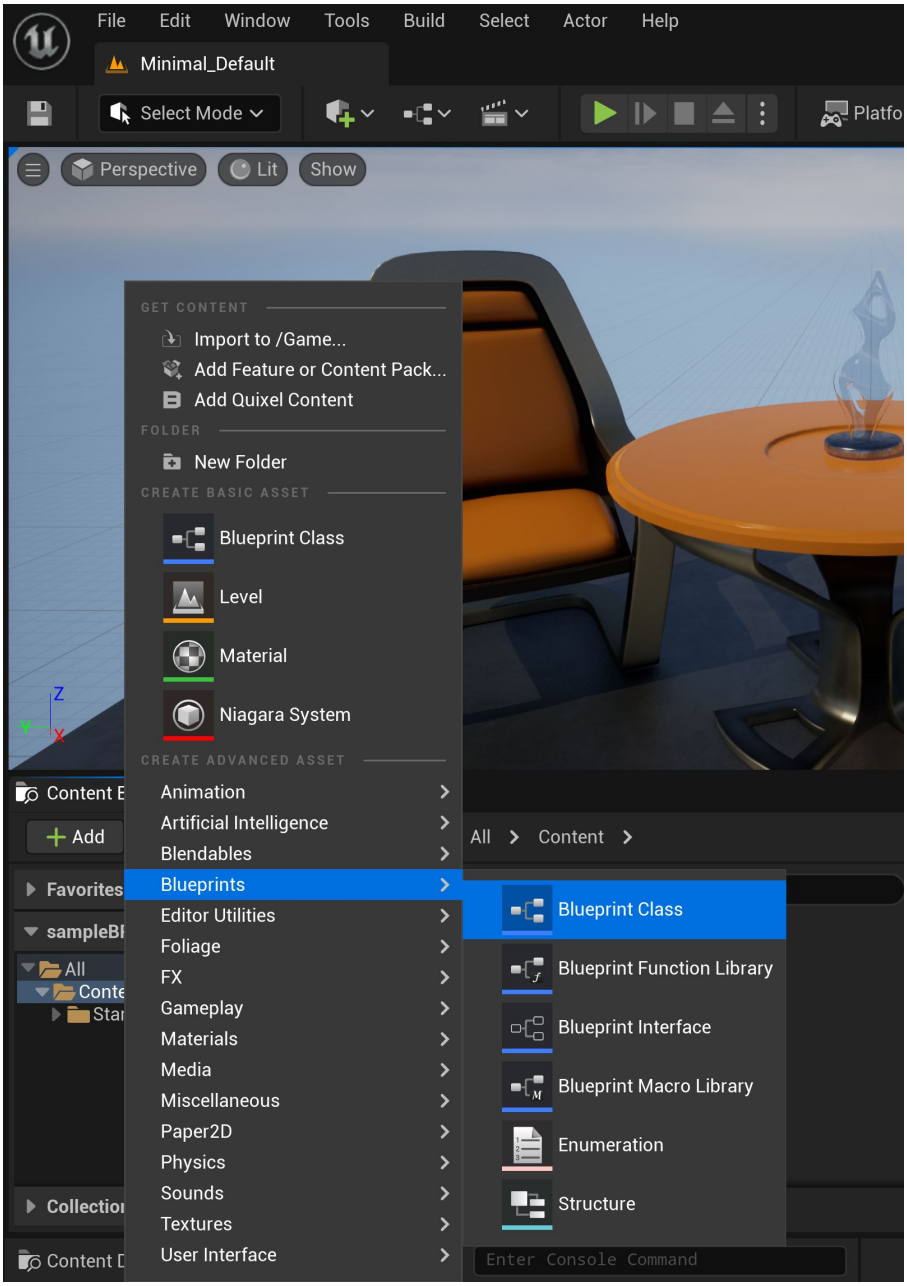


To Trigger Blueprint Values

It's possible to trigger values from Viz Arc into Unreal's Blueprint. This section describes the procedure step-by step. In this sample, an integer value is set from Viz Arc and displayed in the render output of the Unreal Engine.

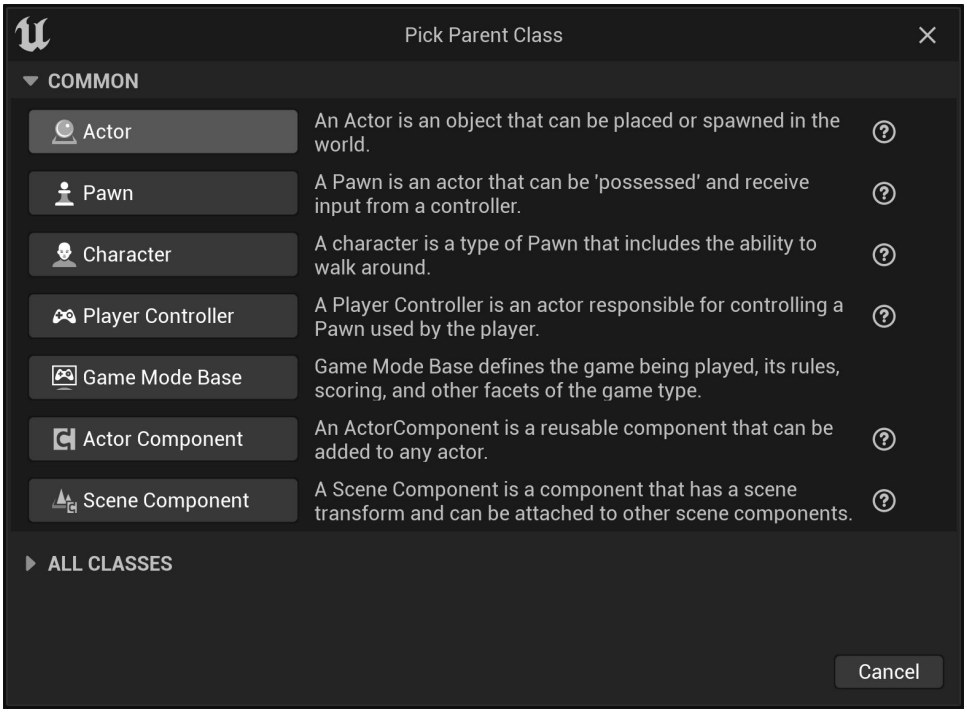
- First, add a Blueprint Class to the project content by clicking the **Add** button in the **Content Browser** view, select **Blueprints** and then **Blueprint Class**.

Copyright © 2025 Vizrt

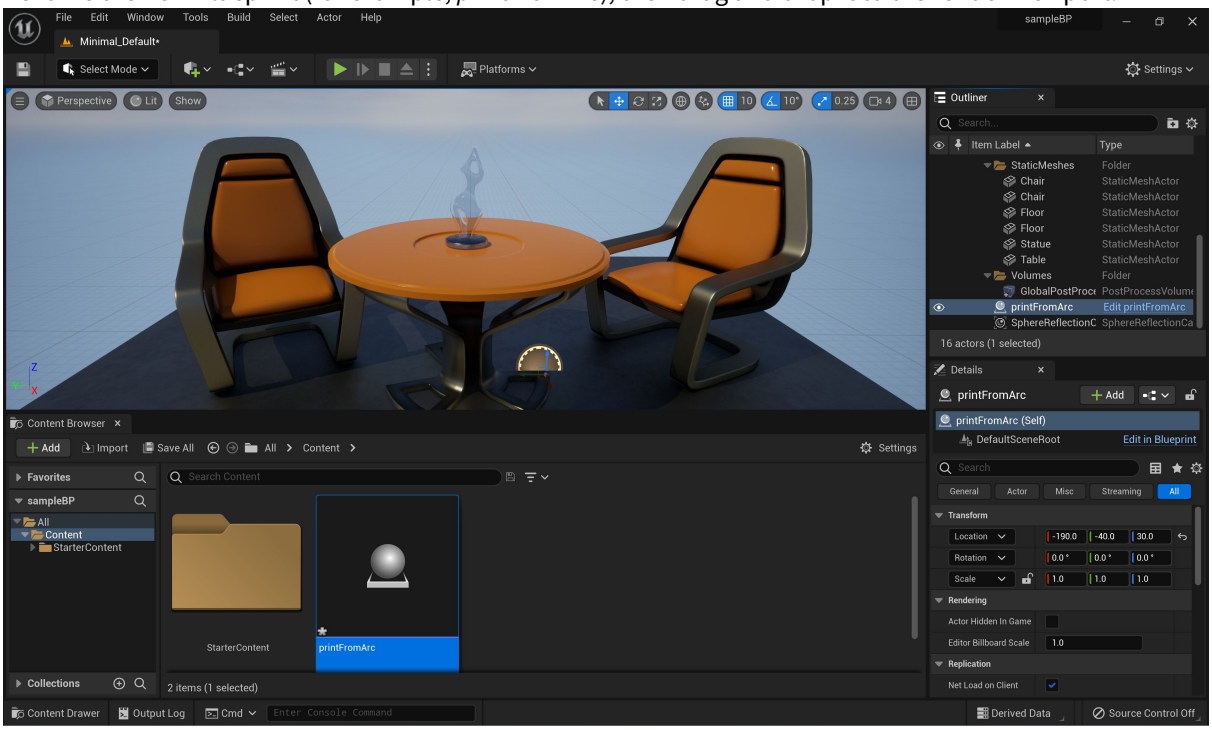


- Select **Actor**.

Copyright © 2025 Vizrt

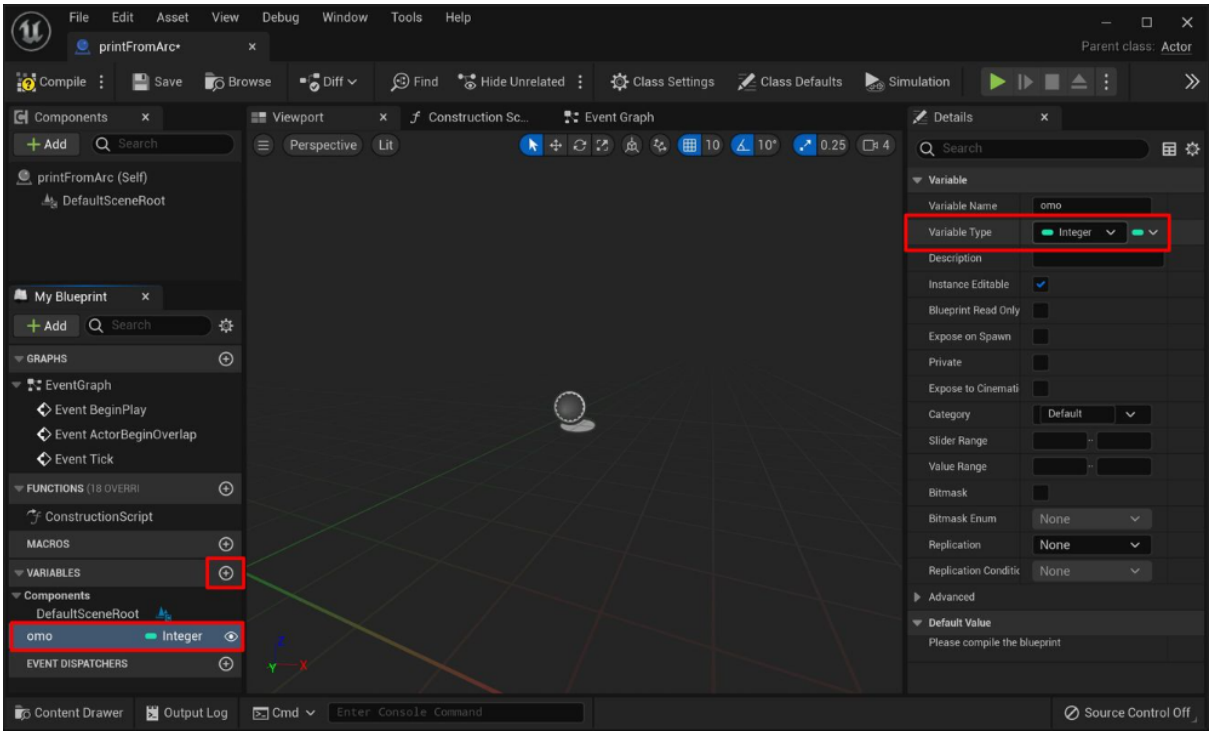


- Rename the new Blueprint (for example, *printFromArc*), then drag and drop it to the render viewport.



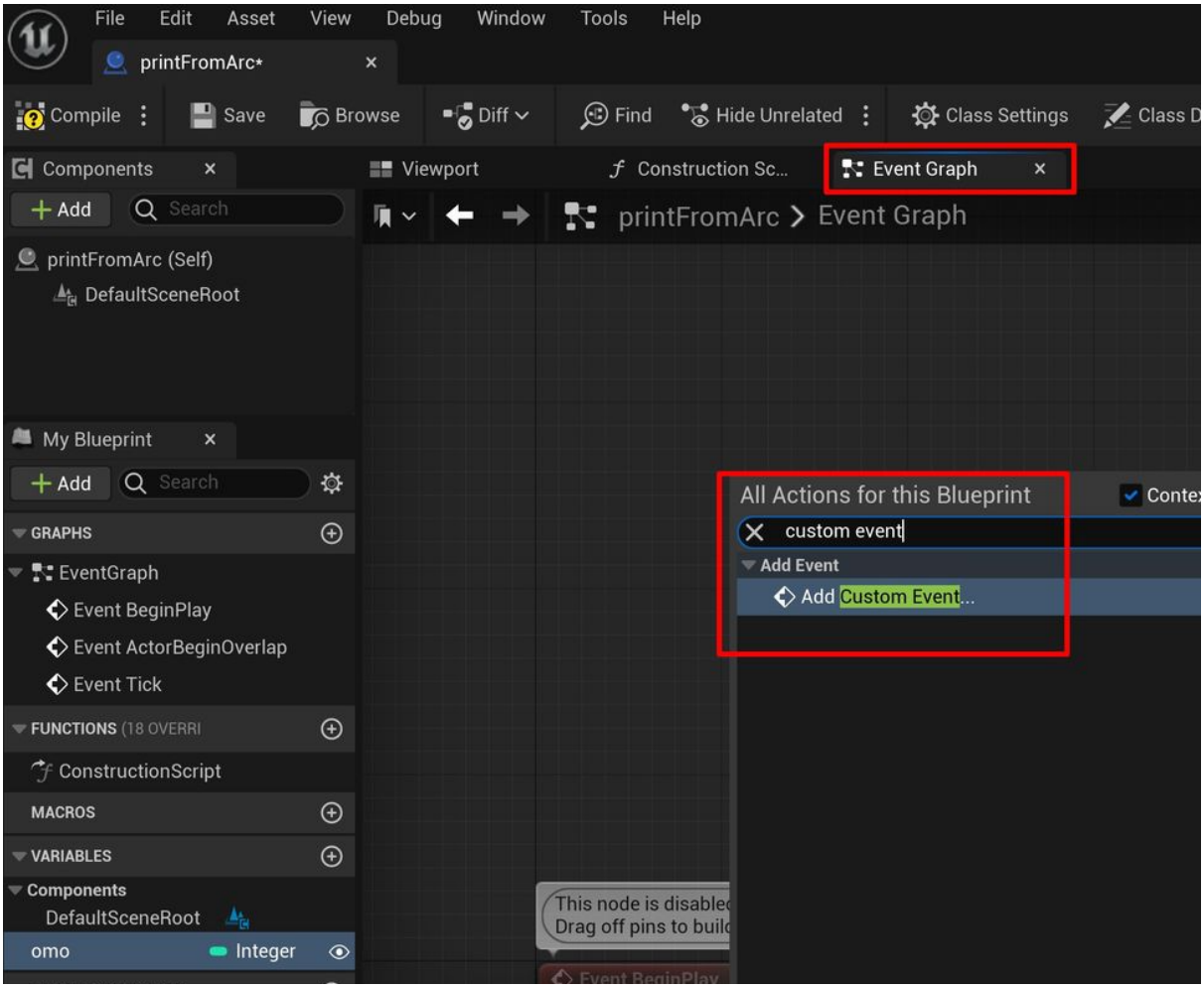
- Open the Blueprint by clicking on the **Edit printFromArc** link in the Outliner view. Create a new variable and rename it to **omo** and set the type to **Integer**.

Copyright © 2025 Vizrt

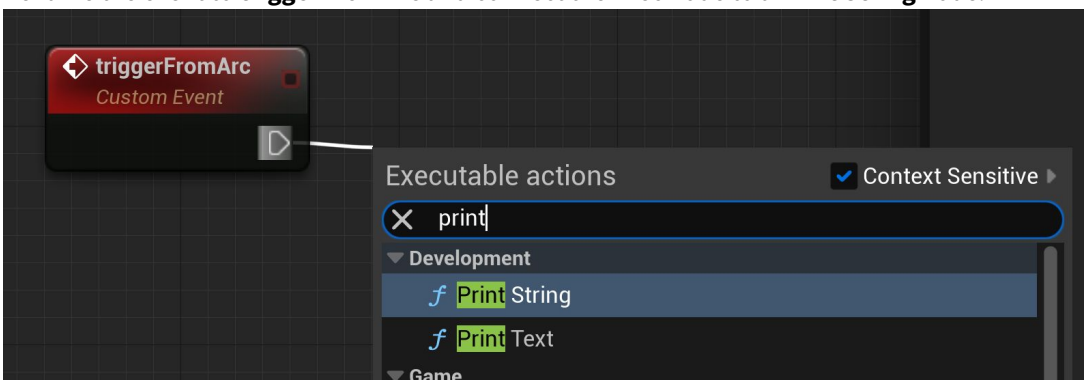


- Select the **Event Graph** tab and right click on the blueprint canvas and create a new **Custom Event.**

Copyright © 2025 Vizrt

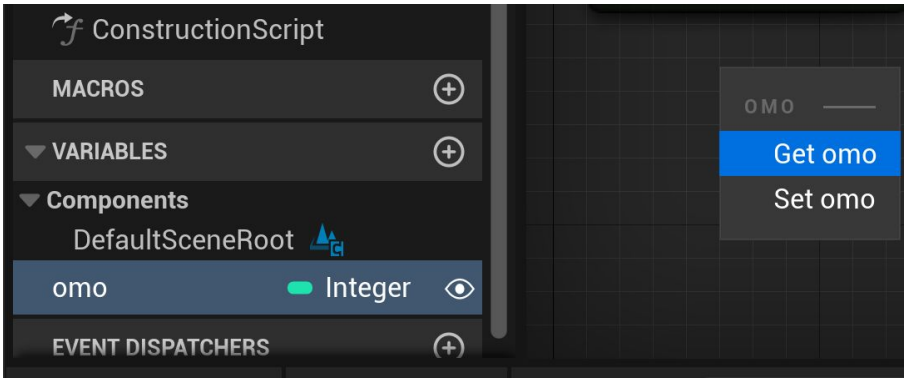


- Rename the event to **triggerFromArc** and connect the Exec node to a **Print String** node.

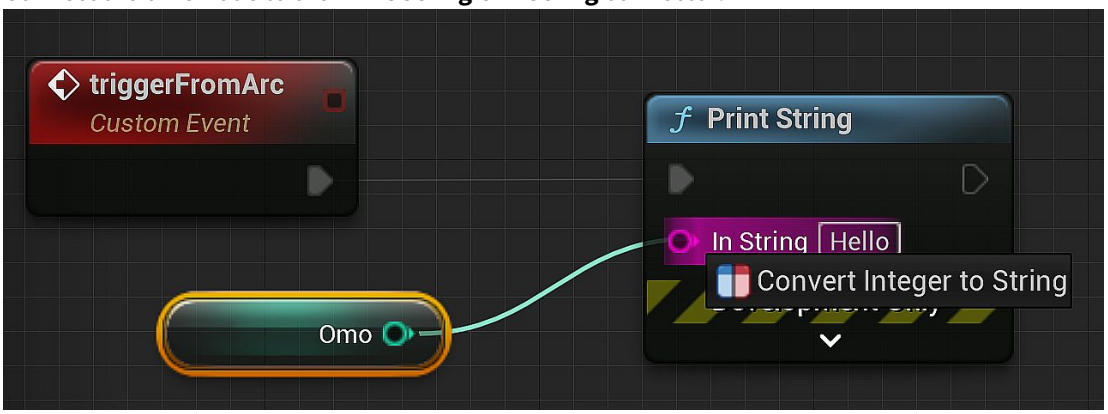


- Drag and drop the **omo** variable to the blueprint canvas and select **Get**.

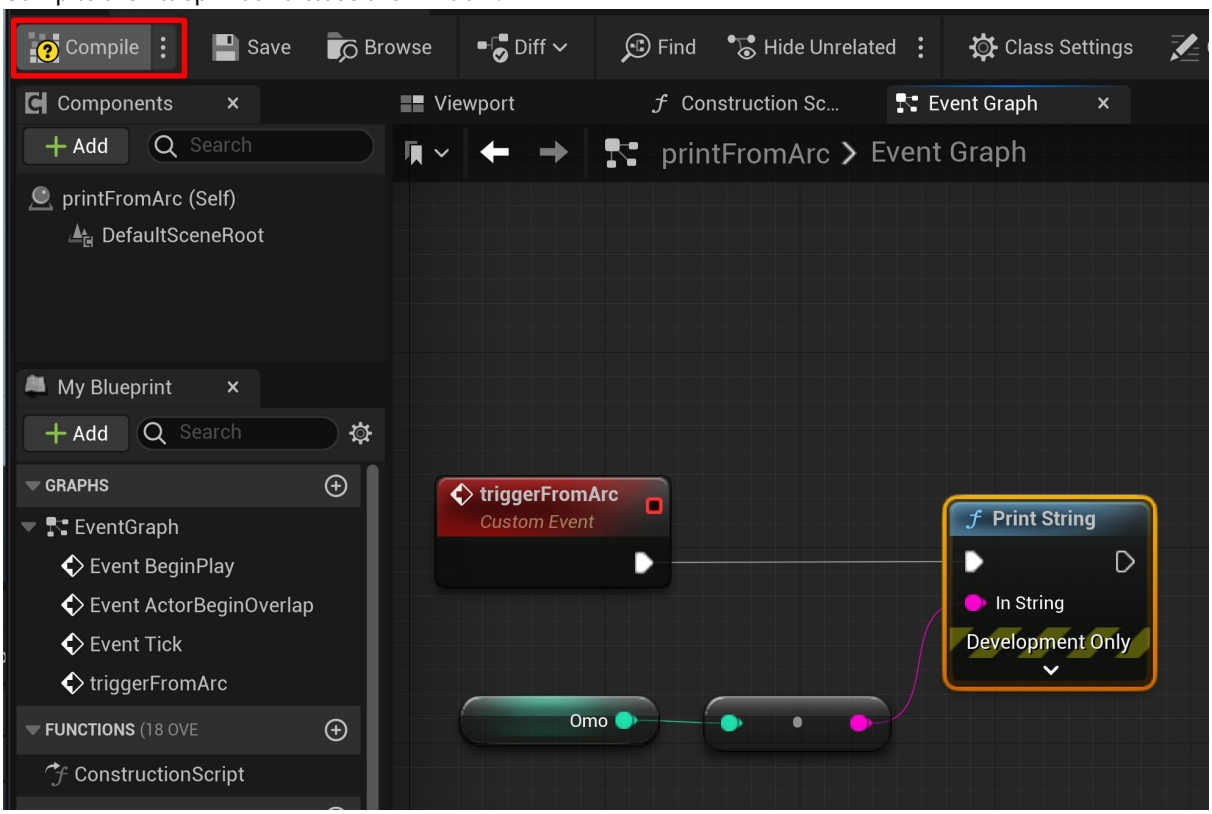
Copyright © 2025 Vizrt



- Connect the **omo** node to the **Print String's In String** connector.

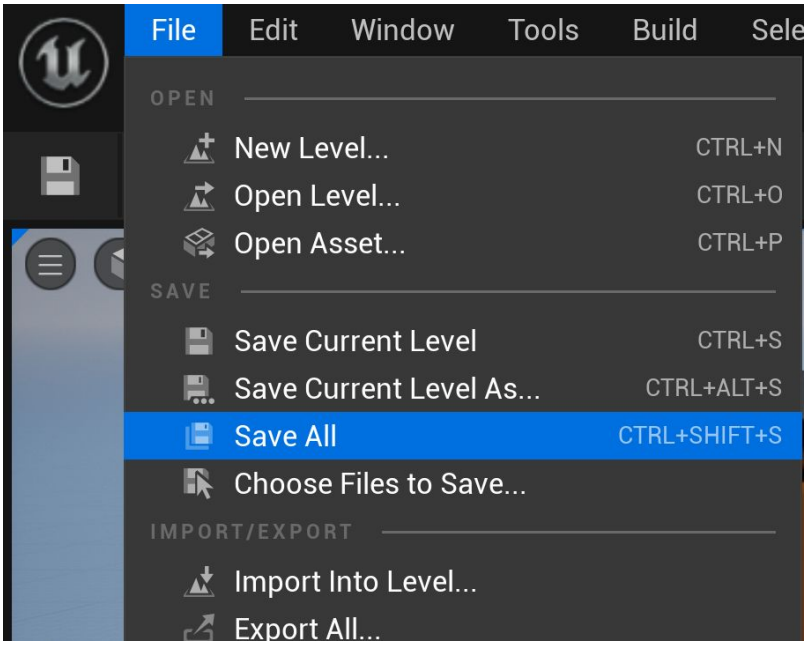


- Compile the Blueprint and close the window.

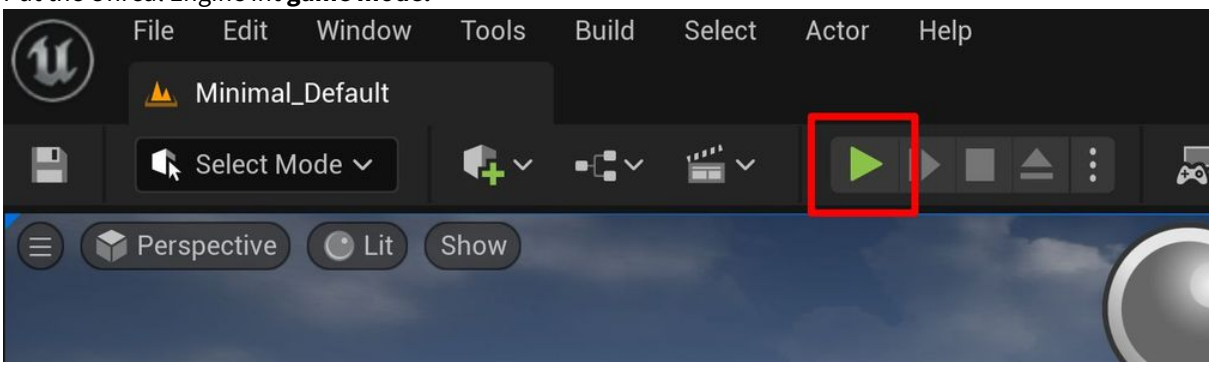


Copyright © 2025 Vizrt

- Save the level.

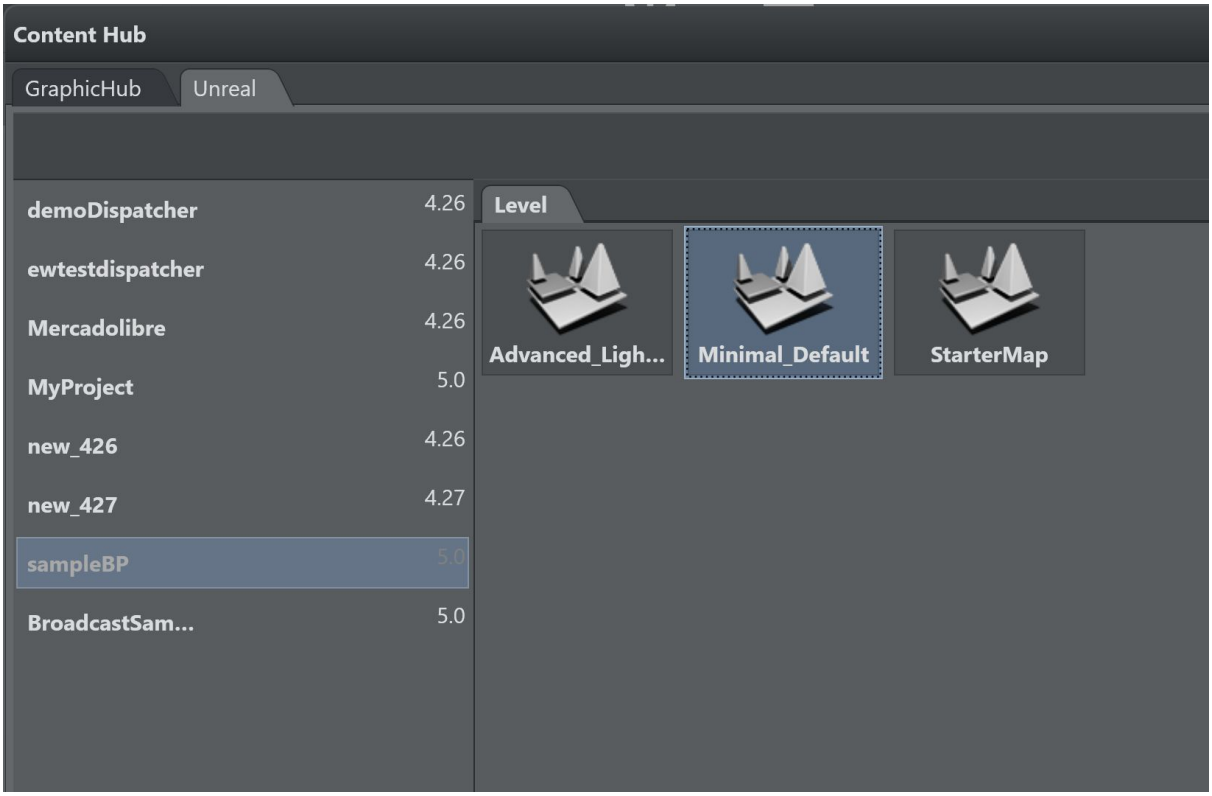


- Put the Unreal Engine int **game mode.**

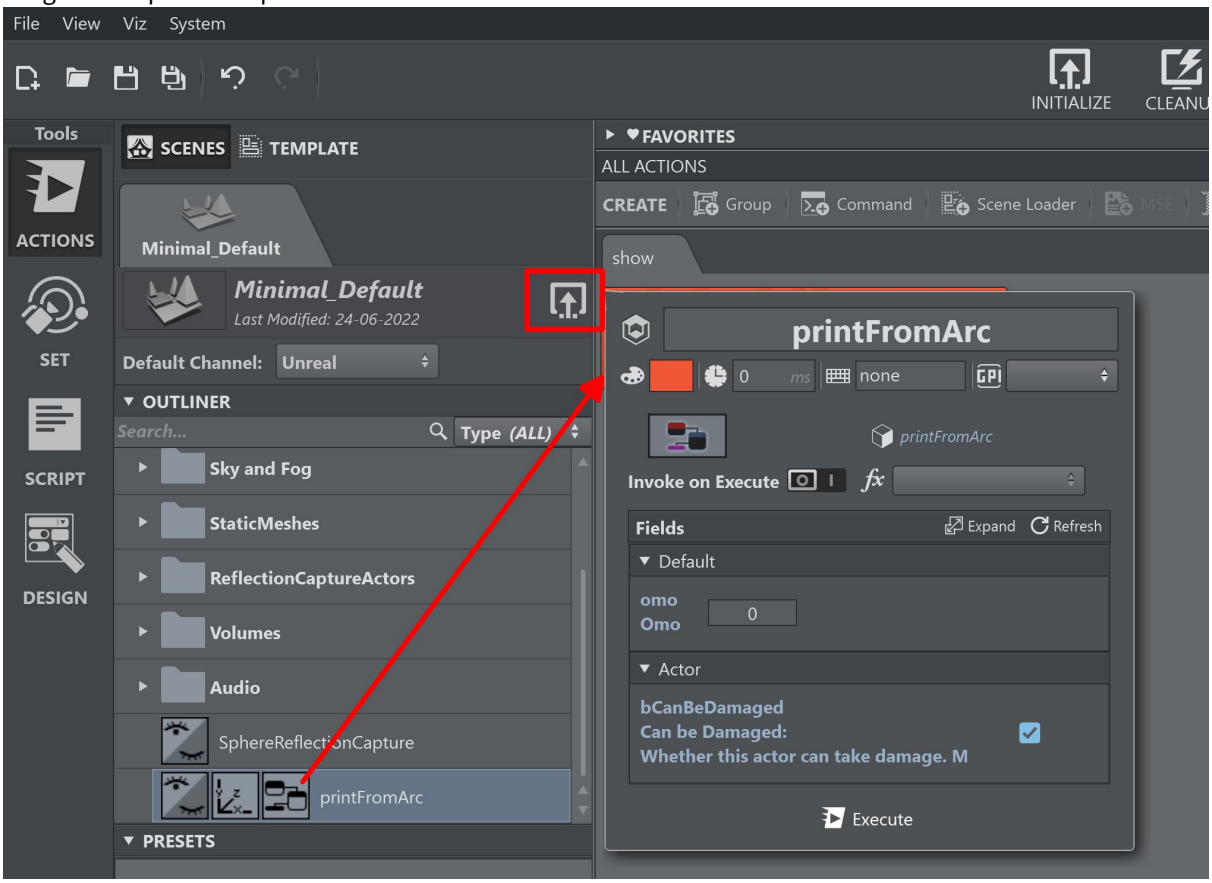


- In **Viz Arc**, select saved level and load the level outliner.

Copyright © 2025 Vizrt

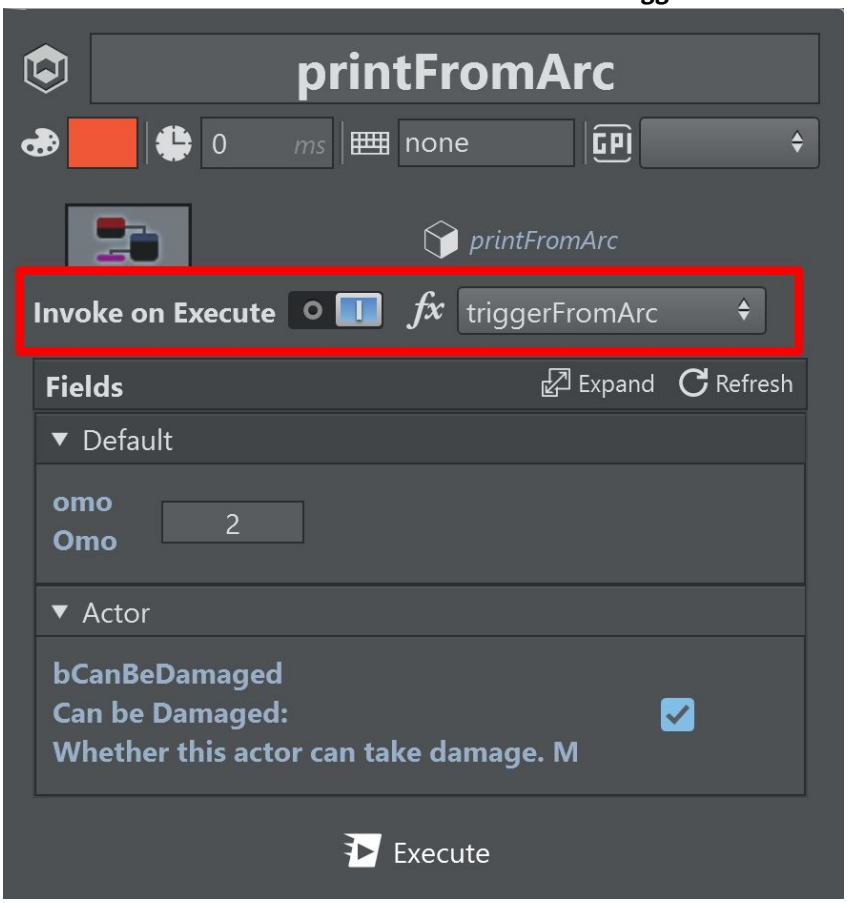


- Drag and drop the Blueprint icon from the outliner view into the action canvas.



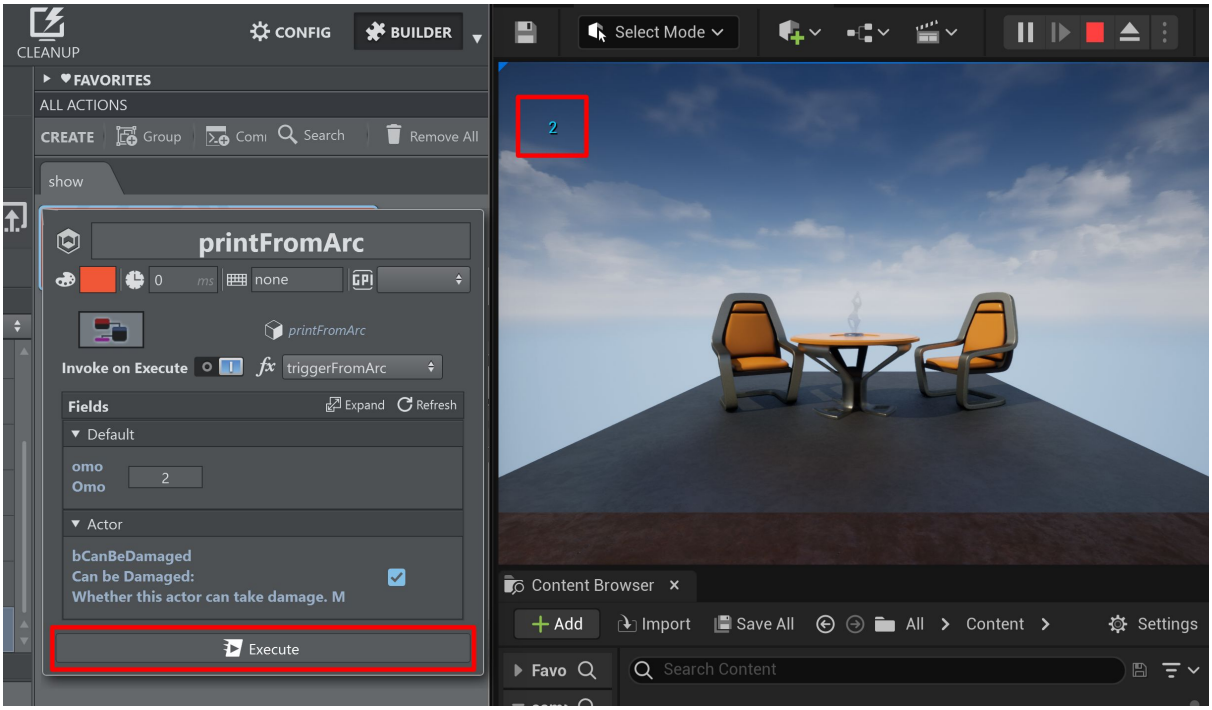
Copyright © 2025 Vizrt

- Check the **Invoke on Execute** checkbox and select the **triggerFromAcr** event from the dropdown.



- Change the omo value to some value and execute the action. The selected number is printed on the render view.

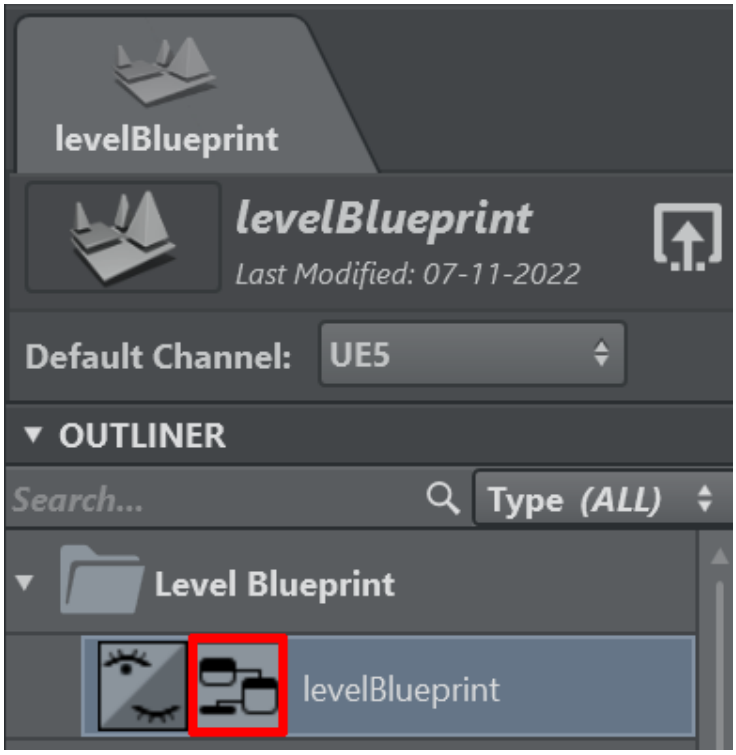
Copyright © 2025 Vizrt



Level Blueprint

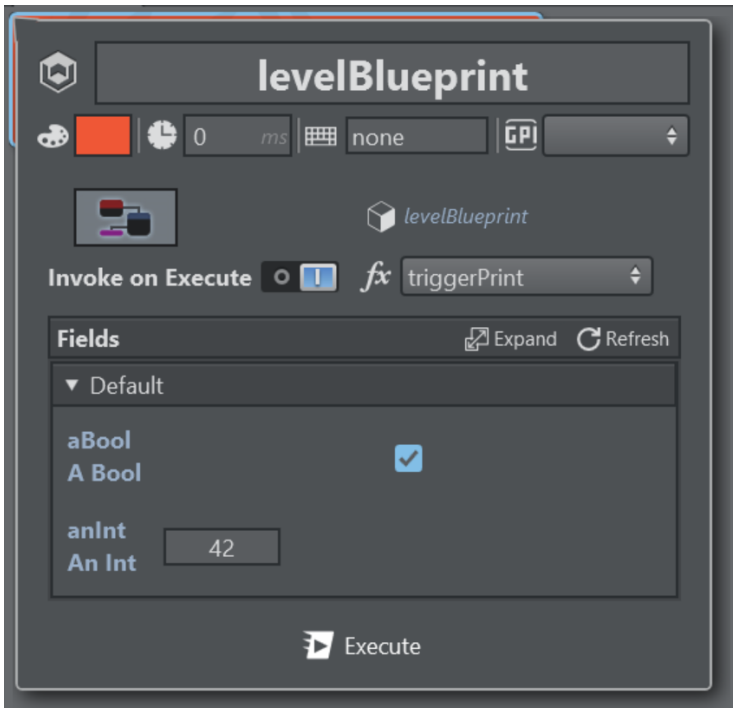
A Level Blueprint is always present on any Unreal level map. The Level Blueprint is visualized in the outliner tree view as the top element under the **Level Blueprint** folder.

Drag the Blueprint to the action canvas to create a regular Blueprint action.



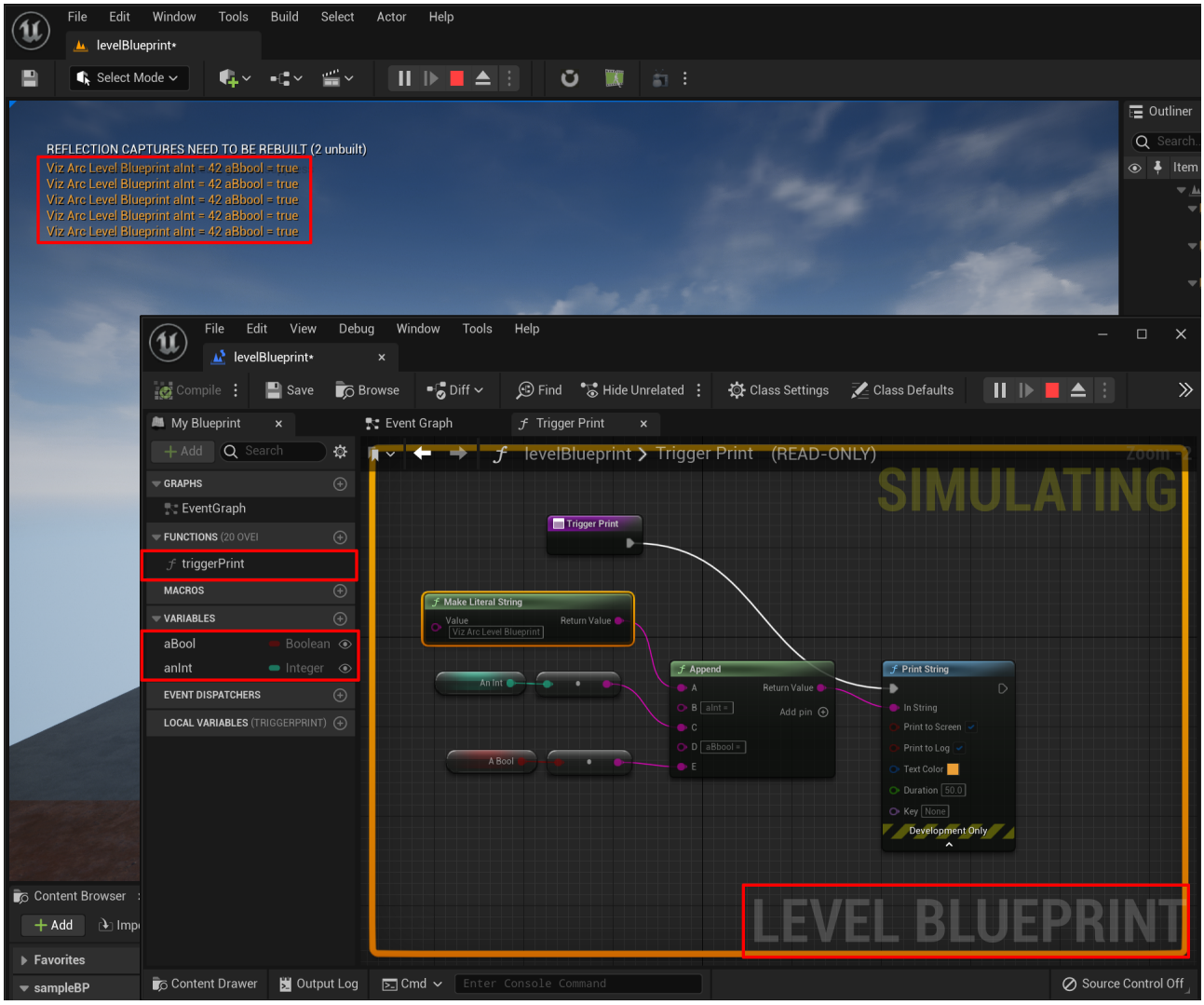
Copyright © 2025 Vizrt

Select a function to be invoked by the Blueprint action and set some of the exposed values.

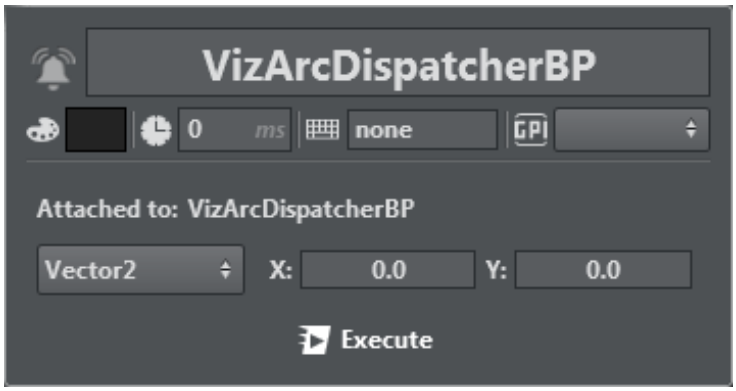


The same blueprint seen on the Unreal Engine which prints a composes string of the given values from Viz Arc.

Copyright © 2025 Vizrt



Dispatcher

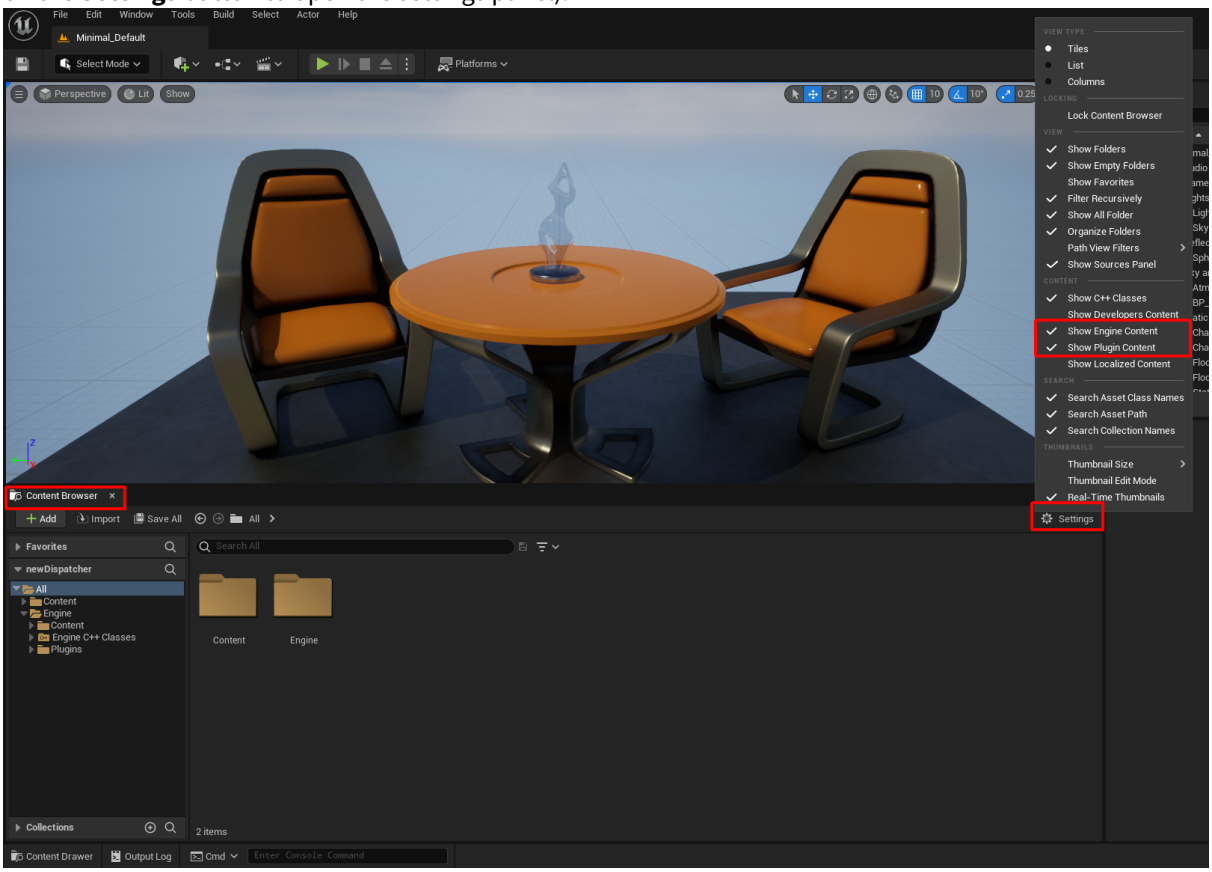


The Dispatcher action lets you create an interaction between Viz Arc and Unreal Engine by setting a value and its type to trigger from Viz Arc to Unreal Blueprint.

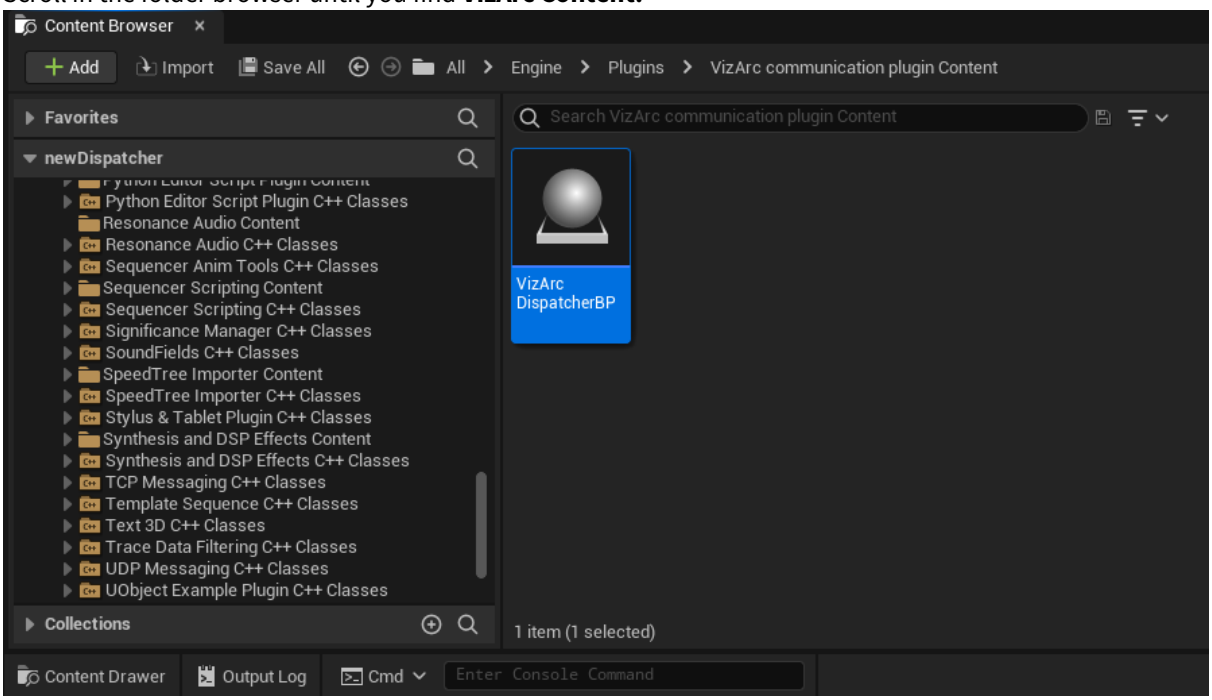
- Enable the **Viz Arc** plugin (if not already enabled during installation).

Copyright © 2025 Vizrt

- Add **Viz Arc Dispatcher BP** to an Unreal level.
- On the content browser, make sure that **Show Engine Content** and **Show Plugin Content** are checked (click on the **Settings** button to open the settings panel).

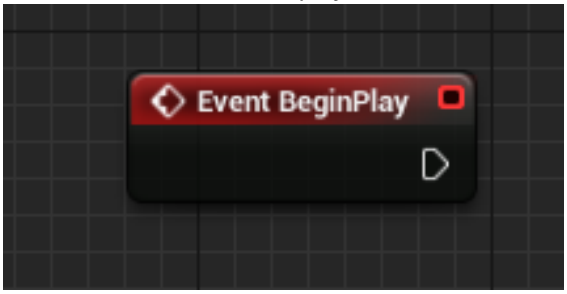


- Scroll in the folder browser until you find **VizArc Content.**



Copyright © 2025 Vizrt

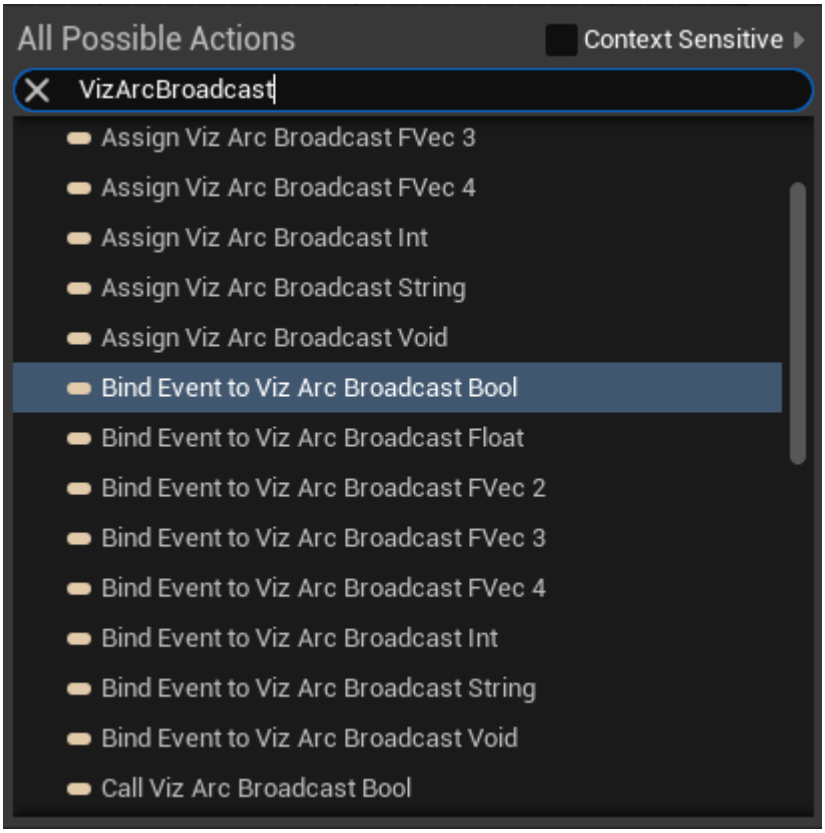
- Drag the **Viz Arc Dispatcher BP** to your level (create actor).
- Subscribe to the dispatcher events (**Viz Arc Dispatcher BP** events).
    - Open or create the Blueprint that is to subscribe to the Viz Arc dispatcher.
    - On the event graph, get the **Event BeginPlay**. This can be done by right-clicking on the canvas, entering `BeginPlay` in the search box and double-clicking on **Event BeginPlay**. This event is called when Unreal enters play mode.



- Once the Begin Play event has been created, you must extend the execution line. Drag the white shape on the right side of the Event shown above. Once you release the left cursor button (stop dragging) the **All possible options** popup appears.

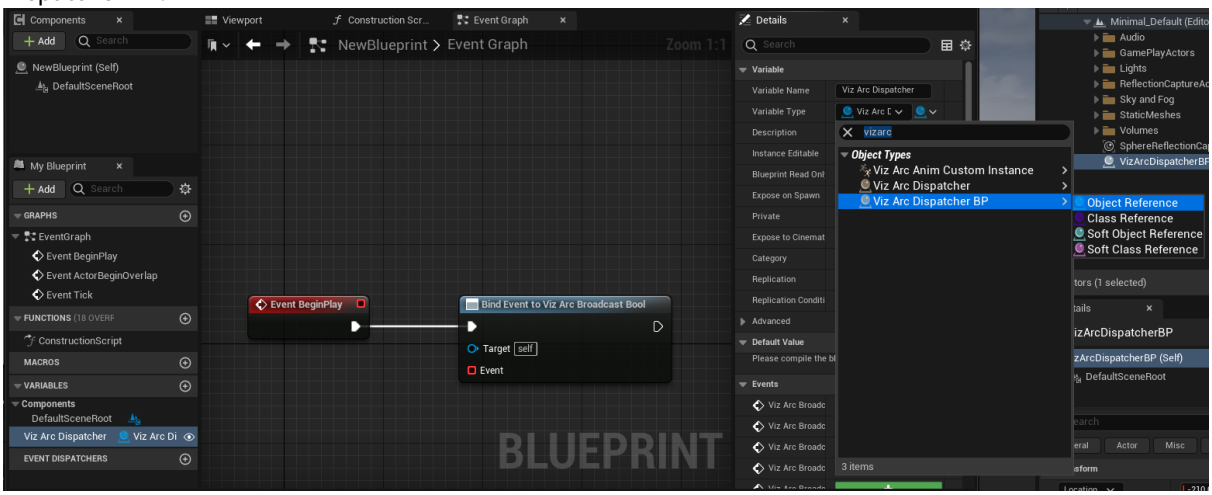
> ⚠ **Note:** To see all Viz Arc Dispatcher BP events, disable **Context Sensitive** in the popup.

- Enter `VizArcBroadcast` in the search box to see all the functions of each available subscription type (**void**, **bool**, **int**, **float**, **string**). To subscribe to the Boolean version, double-click on **Bind Event to VizArcBroadcastBool.**

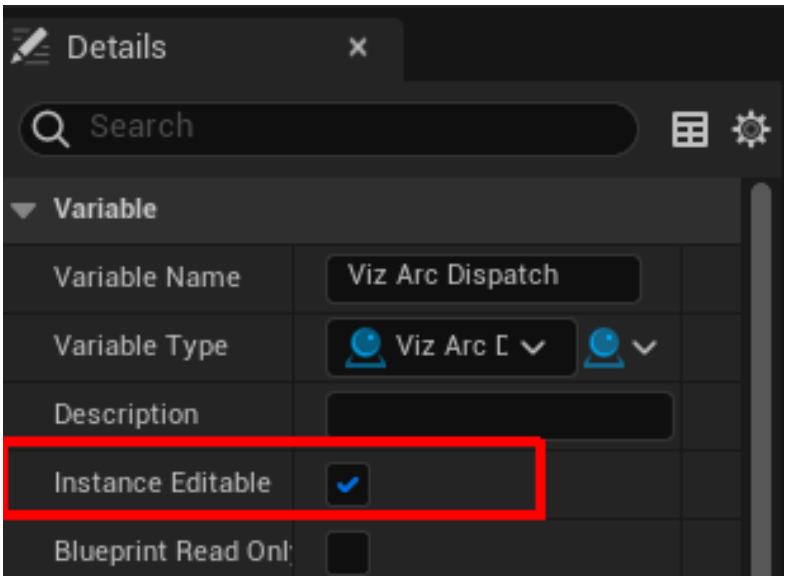


Copyright © 2025 Vizrt

In order for Bind Event to VizArcBroadcastBool to work you need a reference to the Viz Arc Dispatcher BP that you created earlier. To do this, create a new variable of the type **Object Reference** to Viz Arc Dispatcher BP.

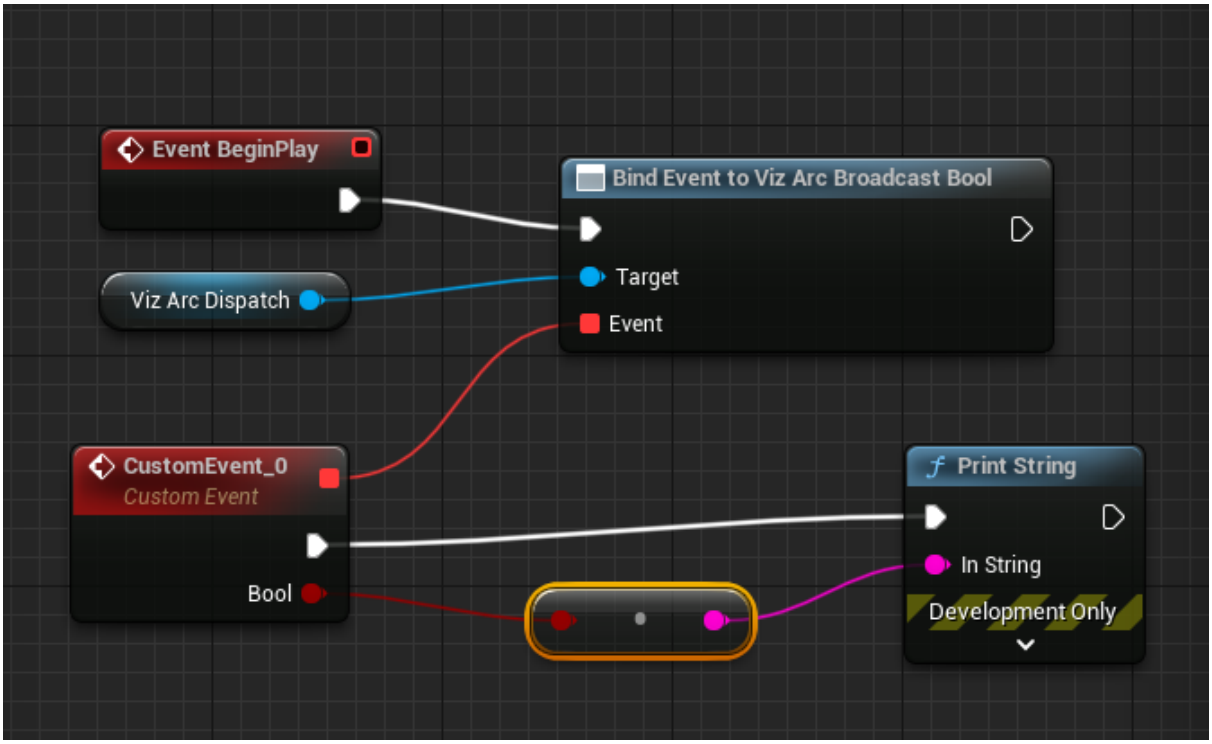


Make it **Instance Editable**:

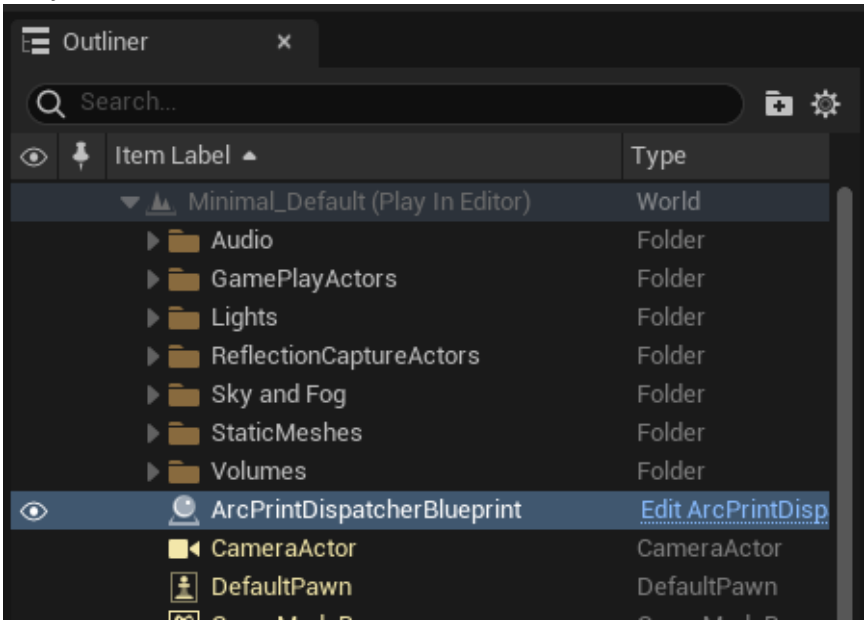


- With the variable created we can now attach it to Bind Event to VizArcBroadcastBool, so that Unreal knows which instance of the dispatcher to listen to. Drag the variable to the blueprint canvas and select **Get**. Connect the node to the **Target Bind Event to VizArcBroadcastBool**.

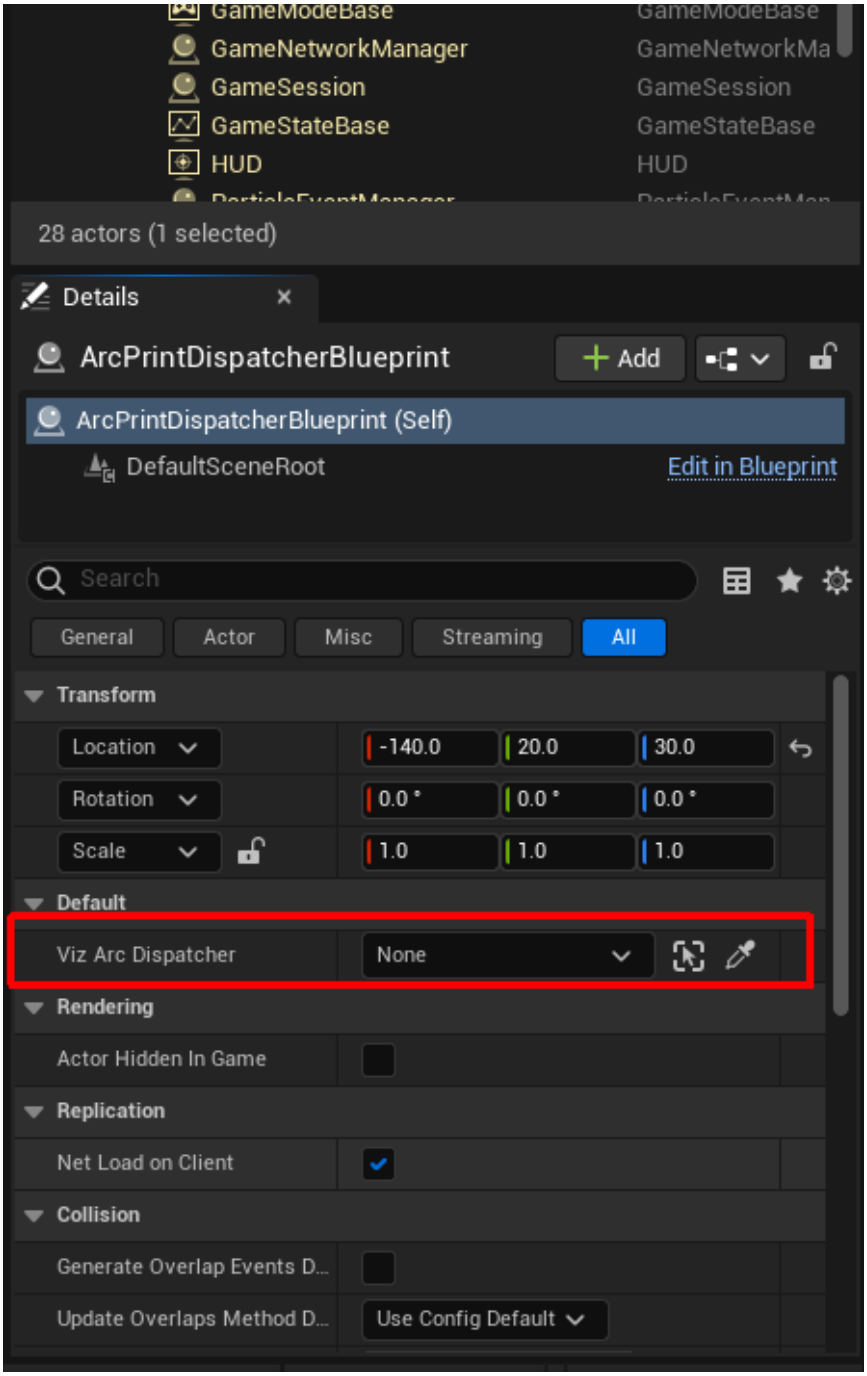
Copyright © 2025 Vizrt



- The only parameter missing is **Event**. To create it, drag the red square on **Bind Event to VizArcBroadcastBool** inside the canvas.
- Release the left cursor button. Once the popup opens, enter **Custom Event** and select the only available option **Add Custom Event...**. This creates the event that is called whenever the dispatcher fires an event of type `bool`. The bool value and event trigger can be used elsewhere in the Blueprint, in this sample just print the bool value on the render output.
- The last step is to set the **Viz Arc Dispatcher Reference**. To do this, go back to the editor, select the **Blueprint** that you were setting up, loop on the details tab for your newly created variable and select the one you want to listen to from the list of **Viz Arc Dispatchers**.

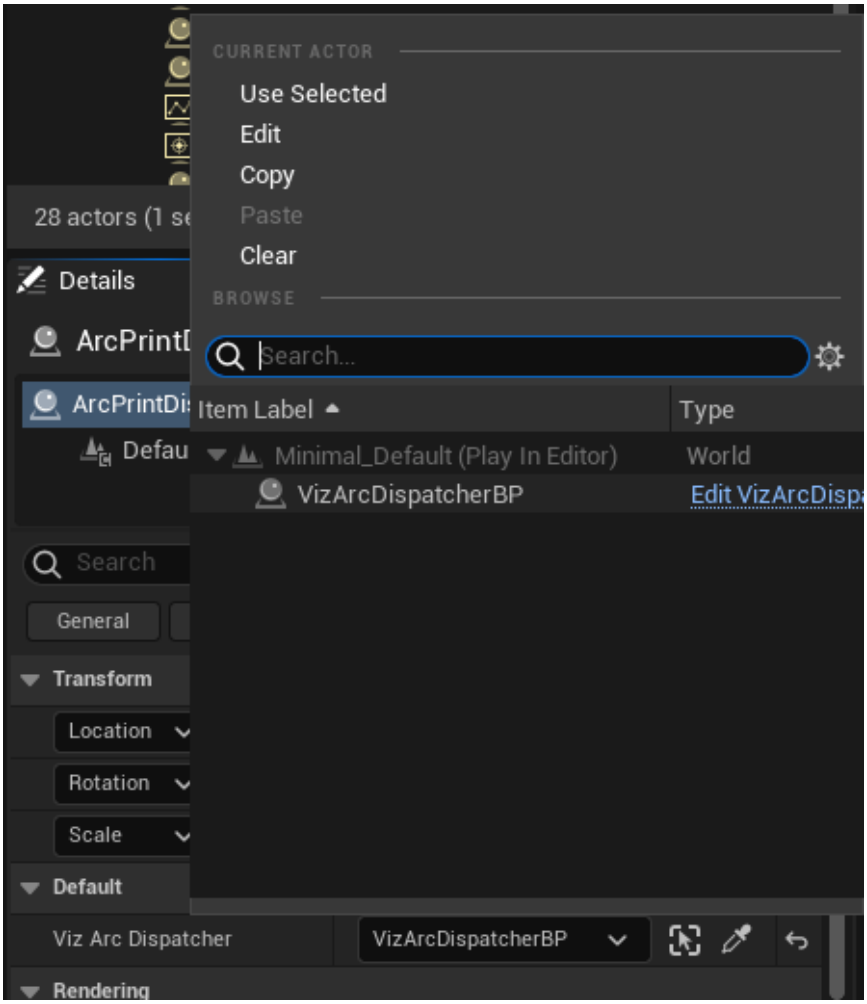


Copyright © 2025 Vizrt



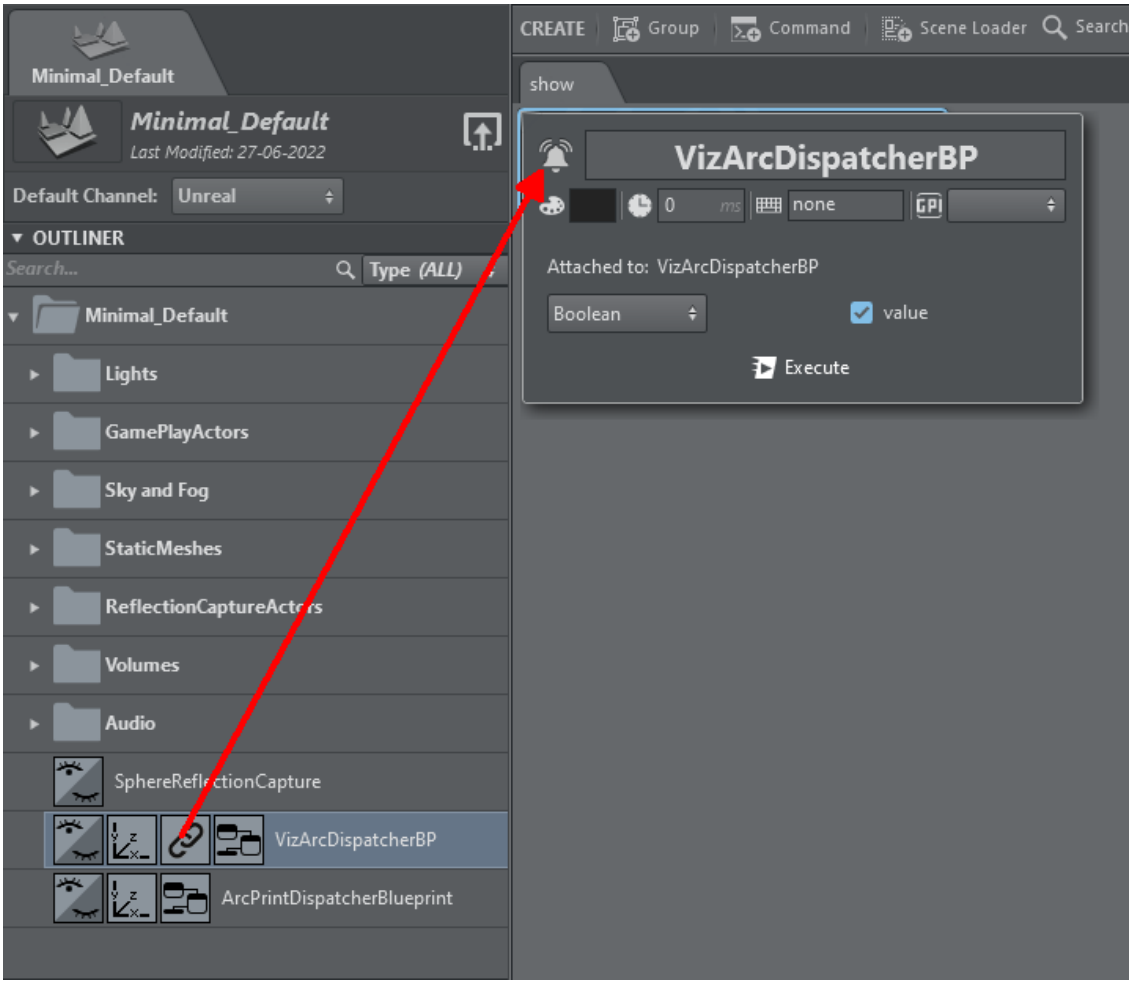
Select the Viz Arc Dispatcher.

Copyright © 2025 Vizrt



- You are now ready to control the variable from Viz Arc.
  - Load a level containing **Viz Arc Dispatcher**.
  - Create a **Viz Arc Dispatcher Action** by dragging the cursor from its icon onto the canvas.

Copyright © 2025 Vizrt



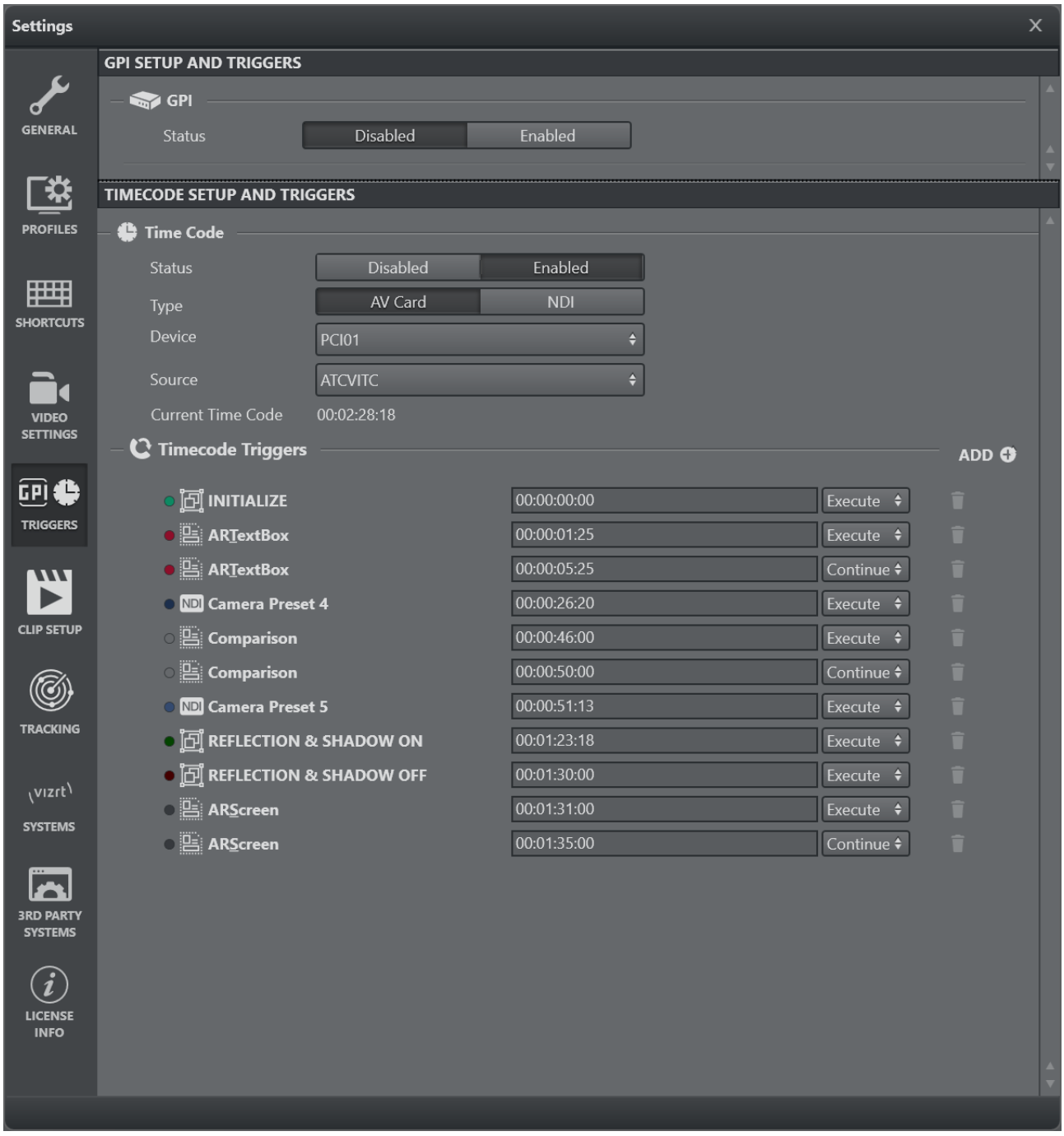
- Click the newly created action's **edit** button. You can use the drop-down to select which type of event you want to fire (**void**, **int**, **float**, **bool**, **string**). Whenever the type changes, the view changes correspondingly so that you can define the value that is sent with the event

Copyright © 2025 Vizrt

5.8.7 Timecode from Plura or NDI

Timecode can be ingested to Viz Arc in two ways. Either through a Plura PCI board, or through a NDI Stream containing Timecode metadata.

Specific timecode can be used to trigger actions, on templates they can execute or continue a template graphics.



- Configuration
  - Adding Actions to a Timecode

Copyright © 2025 Vizrt

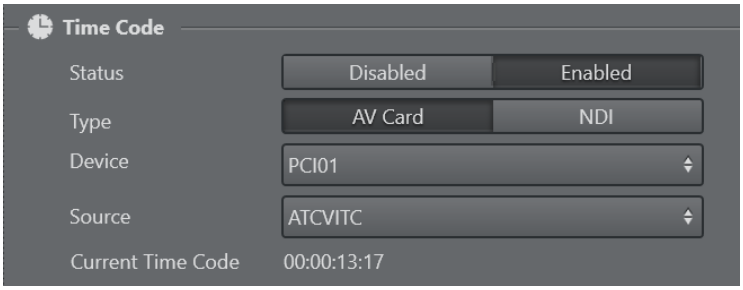
- Using the Action Editor
- DataMap

Configuration

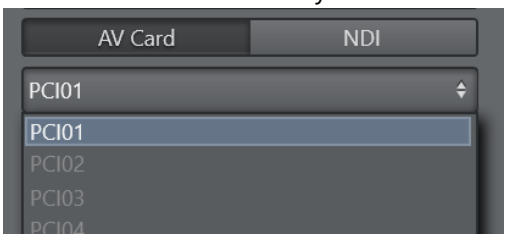
- On the **Configuration** window, select the **TRIGGER** configuration page.



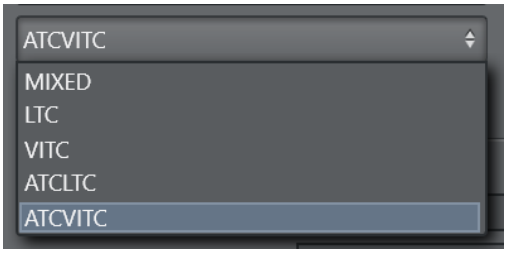
- Under the **Time Code** section, set the Status to **Enabled**.



Select which **Type** of timecode is going to be used. Use an **AV Card** if the system has a Plura timecode reader card installed, and a valid signal carrying a Timecode.
If multiple cards are installed in the system, select the device from the **Device** list. The dropdown only shows the available boards in the system.

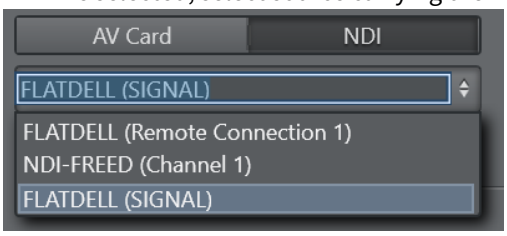


- Select the **Source** where the timecode is embedded within the connected signal.



Choose between one of **MIXED**, **LTC**, **VITC**, **ATCLTC** or **ATCVITC**.

- If **NDI** is selected, select source carrying the Timecode.



Copyright © 2025 Vizrt

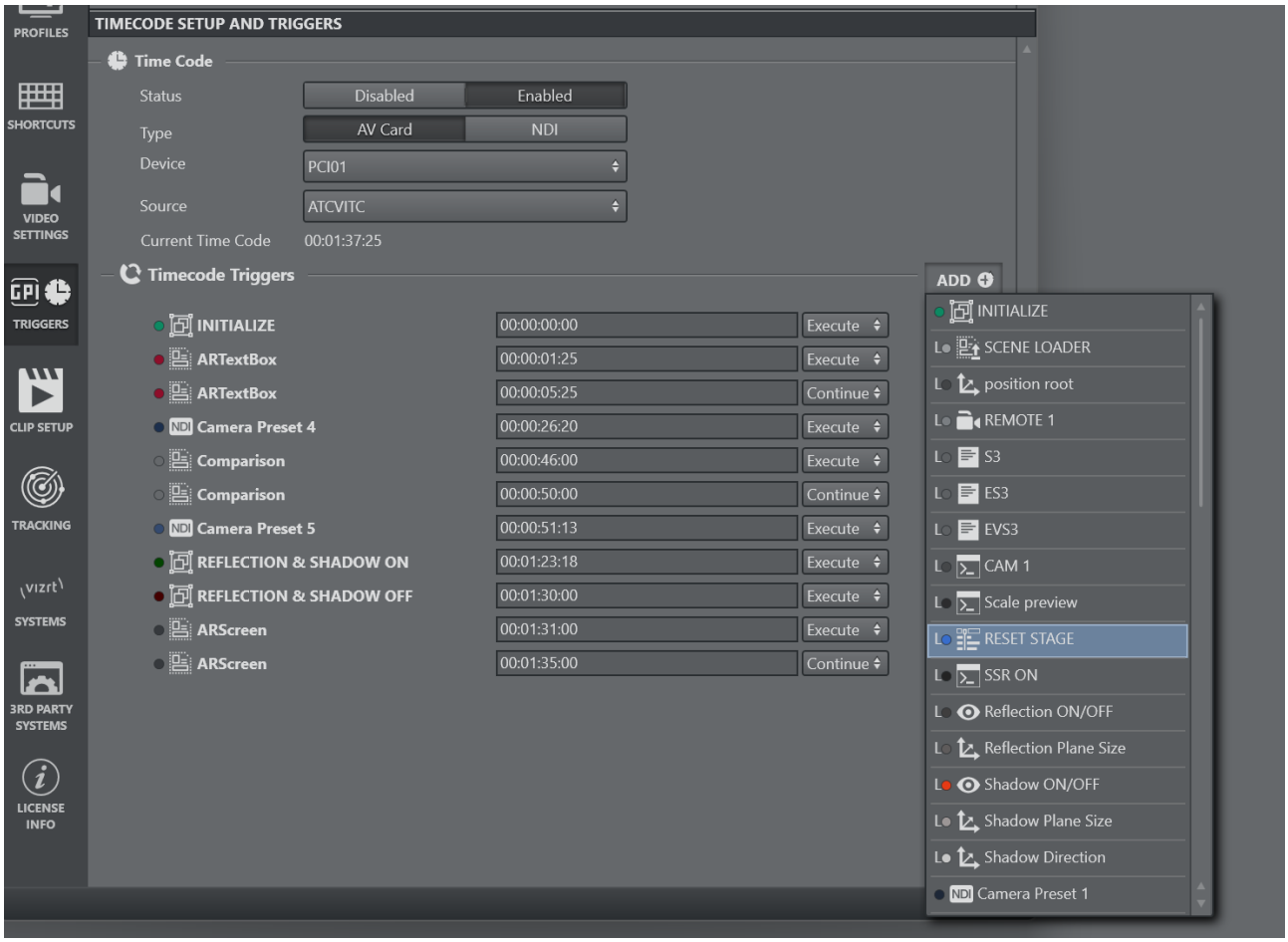
- Once a valid Timecode is read, it is displayed and updated under **Current Timecode** Current Time Code 00:01:04:20, as well as in the bottom status bar 00:01:04:20.

Adding Actions to a Timecode

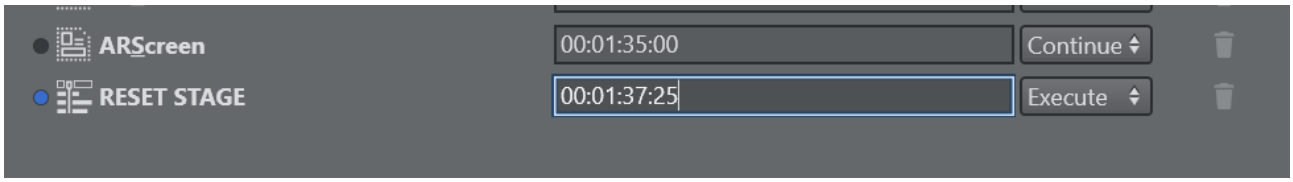
There are different ways to associate a Timecode to the execution of an action. One is on the action editor, and the other on the configuration panel.

Using the Configuration Panel

Click the **ADD** button to choose among one of the actions present in the project. The action is added with the current Timecode.



The action is added to the trigger list with the new Timecode.

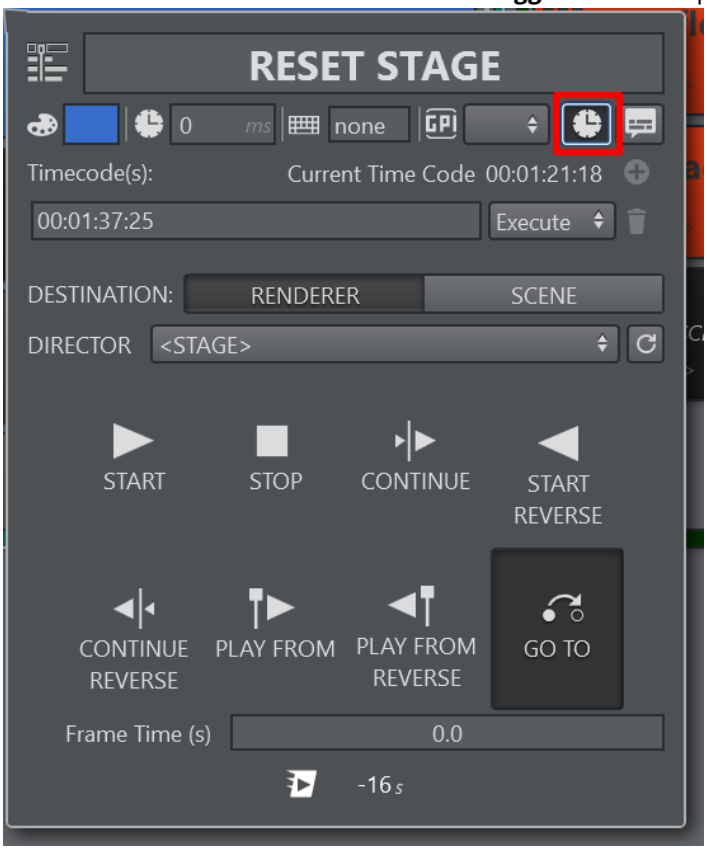


The execution type is **Execute** by default. It can be set to **Continue** and used for **template actions**, in which case, the continue animation is triggered.

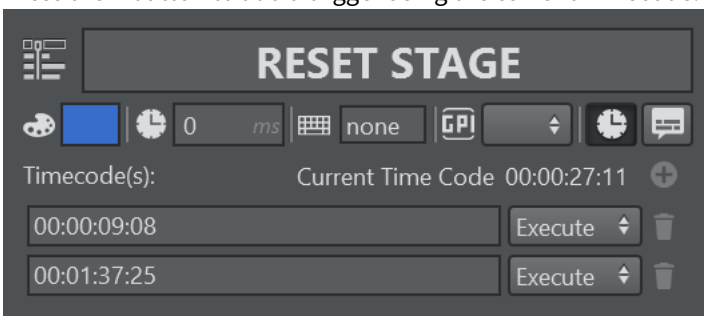
The timecode can be modified manually afterwards, the format is **HH:MM:SS:FF**. **FF** represents frames or fields, and it can be between 0 and 24, or in other cases, between 0 and 59 representing field or frames of a 60Hz stream.

Using the Action Editor

- Edit the action and click on the **Time Code Trigger** button to open the Timecode editor.

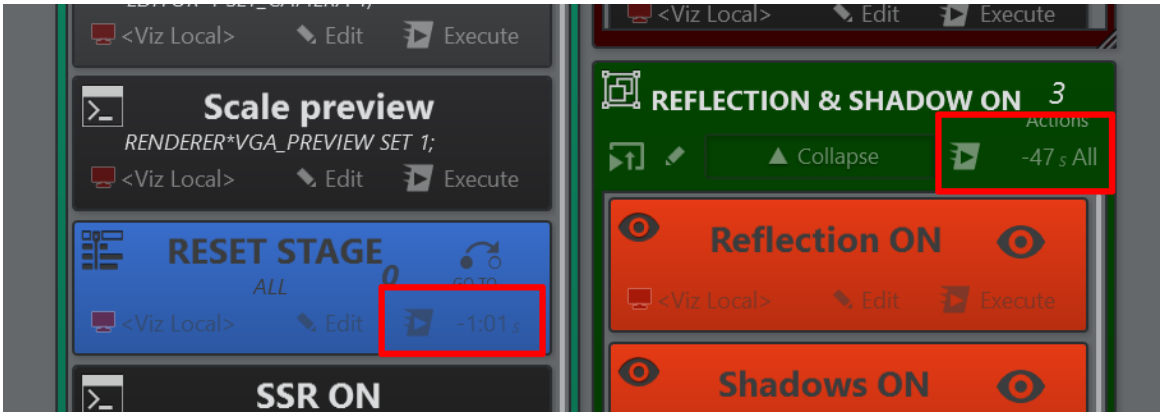


- Press the **+** button to add a trigger using the current Timecode.

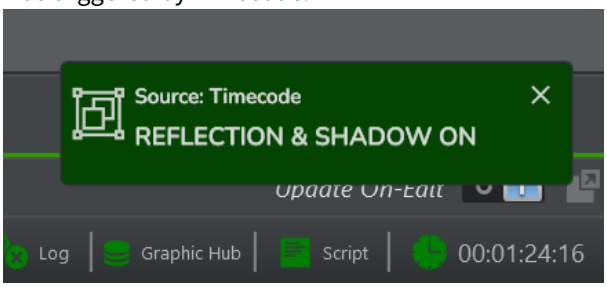


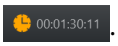
  The Timecode is inserted in an ordered list.
- A countdown on the execution buttons shows the time remaining, until the respective action is triggered.

Copyright © 2025 Vizrt



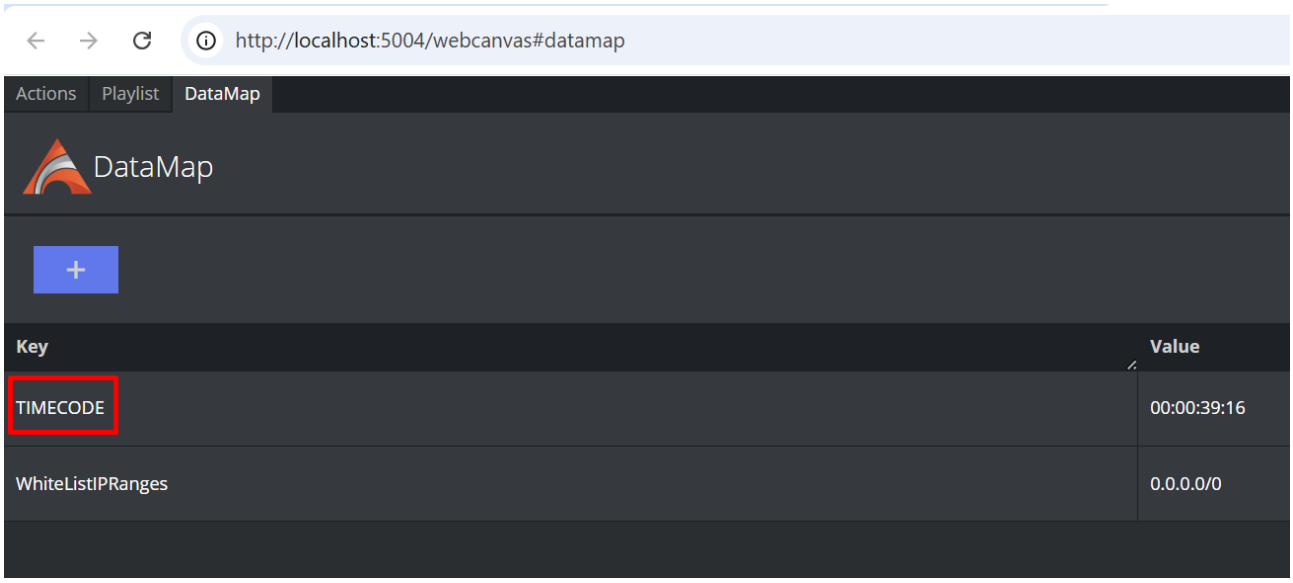
- If the **Toast Notifications** are enabled, a notification is show on the bottom right, showing that an action was triggered by Timecode.



- The trigger mechanism can be disabled by clicking the timecode icon on the profiles bar, changing to yellow when disabled  .

DataMap

The current timecode is stored as a **TIMECODE** key in the DataMap, and can thus be used in templates if necessary.



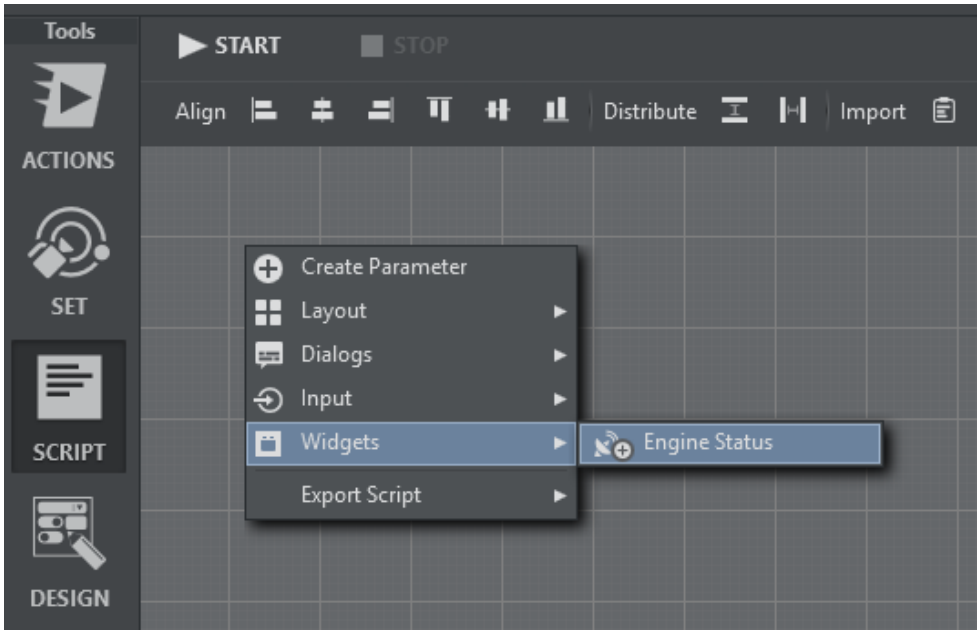
Copyright © 2025 Vizrt

5.9 Engine Status Widget

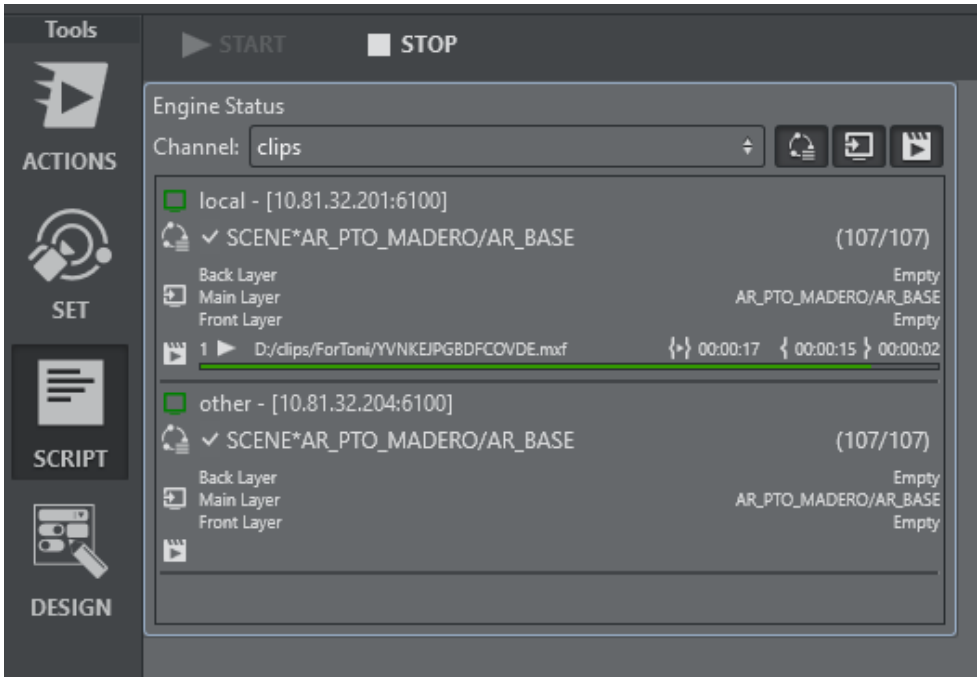
The engine status widget shows you the status of the profile's Viz Engines, including information about the currently loaded scene of an engine, what the engine is currently loading, and clip status.

5.9.1 To Add an Engine Status Widget

Go to the main script (or any template script), right click on the UI editor and select **Widgets > Engine Status**:

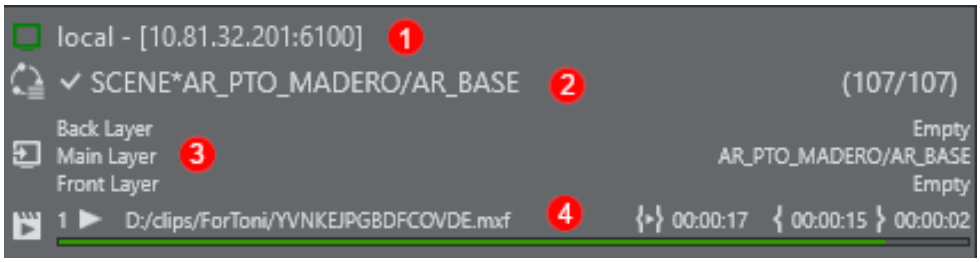


Resize and position the widget. Start the script and select a **Channel** in the drop down menu:



Copyright © 2025 Vizrt

The widget displays a list of Viz Engines for the selected Channel:

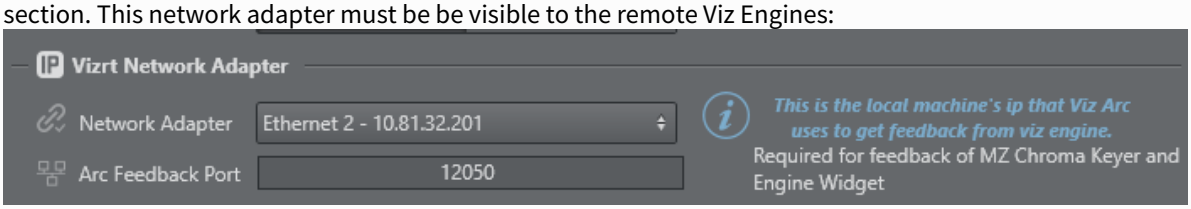


1. **Engine Status**: Shows the Viz Engine Address and Port and the status (green = On Air, orange Off Air, red = not connected).
2. **Load Status:** Shows the last loaded element on the engine. A ✓ at the beginning indicates that a scene has finished loading on the engine, while it's loading it shows a spinning circle ↻.
3. **Layer Status:** Shows what's currently loaded on the three principal layers.
4. **Clip Status:** Shows a list currently active clip channels of Viz Engine and it's respective playback status including total time, current position and time to the end of the clip.



In addition to the Channel dropdown menu, you can also use the toggle buttons to hide or show **Load Status**, **Layer Status** or **Clip Status**.

Note: To function properly, an appropriate **Network Adapter** must be configured Vizrt Systems config section. This network adapter must be be visible to the remote Viz Engines:



Copyright © 2025 Vizrt

5.10 Graphic Hub Browser

This section guides you through the functionalities of the **Graphic Hub Browser**, a web-based tool integrated in Viz Arc that allows you to browse for assets and media in the Graphic Hub.

This new, web-based, tool replaces the previous Browser in the endeavor of resolving performances issues and adding new features.

> **Info:** You can go back to the previous version of the Graphic Hub Browser by **switching off** the button Enable Graphic Hub's web browser under the *General* section of the Settings.

The Graphic Hub Browser shows Folders and Assets by querying the Viz GH REST specified in the Vizrt Systems Configuration section of Viz Arc settings.

Copyright © 2025 Vizrt

5.10.1 Main Features



- Projects and Folders Tree
- Folder's Content / Assets
- Assets Type
- View Mode / Sorting
- Search
- Bookmarks
- Text Filtering
- Import Images into the Graphic Hub

Copyright © 2025 Vizrt

Projects and Folders Tree

The left panel illustrates the **Projects** and **Folders** structure of the Graphic Hub. Click on a folder's name to see its content. You can explore the subfolders by clicking on the arrow icon  or by double-clicking on the folder's name.

Folder's Content / Assets

Once a folder is selected, the content is displayed according to the **Asset Type** selected. Once you've found your asset, you can select it by double clicking.

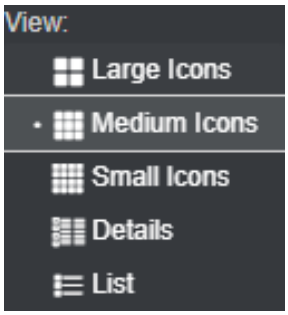
Assets Type

Here, you can switch between various Assets Type defined by the Graphic Hub, each one represented by a tab. You can only browse one kind of assets at a time. The available types are shown according to the Asset you are looking for. For example, if you are looking for an image, solely the **IMAGE** tab is available.

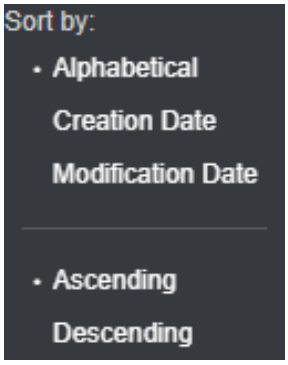
View Mode / Sorting

Here you can choose the display options of the folder's content.

Clicking on the view-mode button  you can choose how the Assets are displayed.



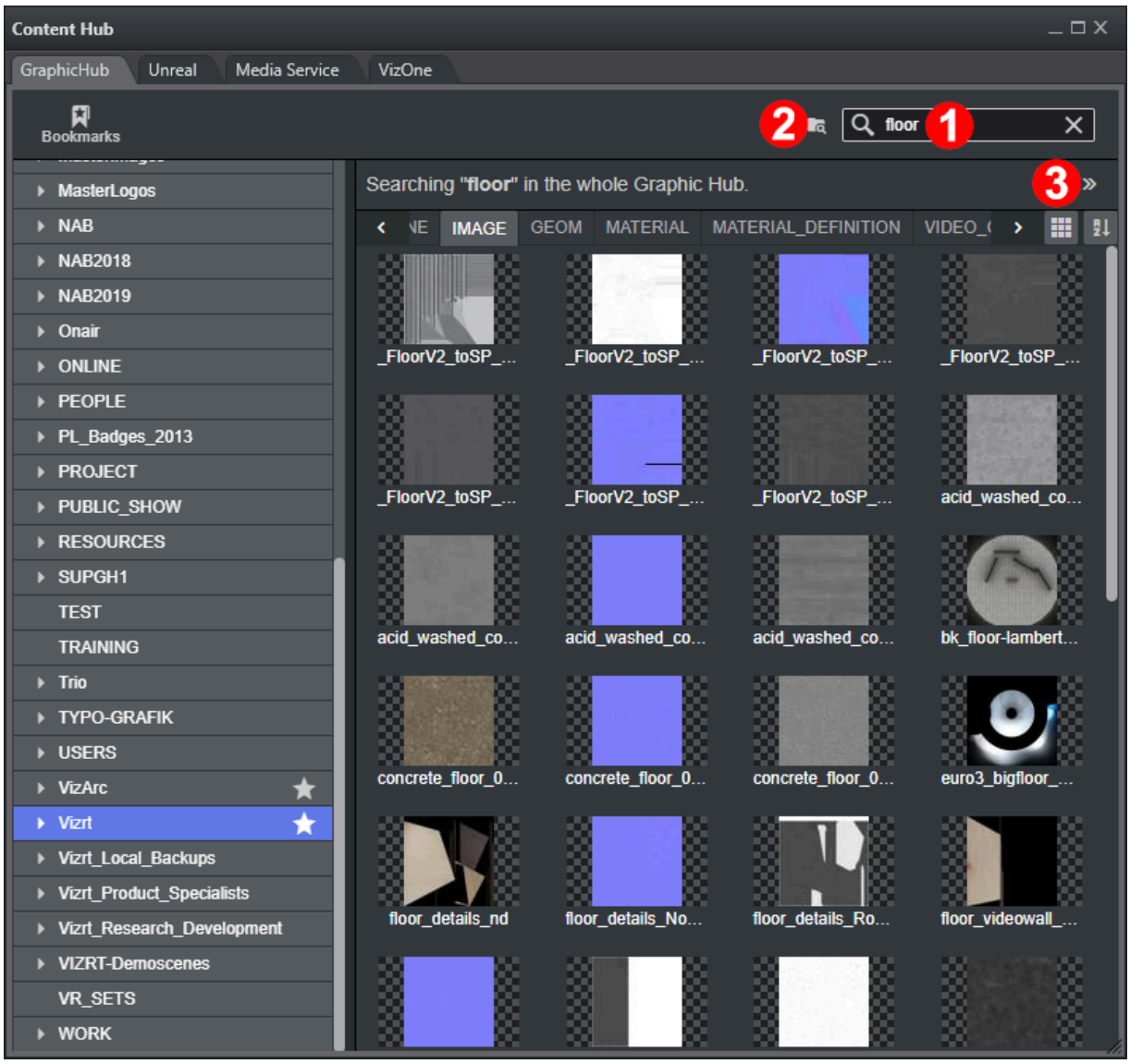
Clicking on the sort button  you can choose how the Assets are sorted.



Copyright © 2025 Vizrt

Search

Here you can search for Assets in the Graphic Hub. Click the magnifier icon, and a text field expands and shows you the search widget.



In the text field marked with **1** you can type the filename (or part of it) of the file(s) that you are looking for. When you press enter or leave the text field, the query starts and the search results are shown.

> **Info:** Like the Folder's content, also the search results can be displayed and sorted according to the user preference.

By default, the search is made **Graphic Hub wide**, meaning the GH REST looks for the files with the specified filename in every Project of Folder, regardless of where they are located in Graphic Hub.

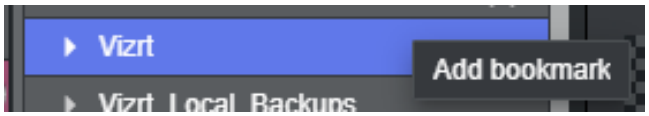
Copyright © 2025 Vizrt

You can **restrict the search to the current folder and subfolders** by clicking on the folder icon marked **2**. When this switch is enabled, you can select any folder on the **Projects and Folders tree** and the search is restricted to the newly selected folder and subfolders.

You can close the search results by clicking on the double chevron icon marked **3**. Whenever you click again on the search button, the last query result is shown.

Bookmarks

You can bookmark every Folder in the Graphic Hub. Right-click on the desired folder, a context menu shows up, then click on the **Add bookmark** option.



You can now open a previously saved bookmark by clicking on the **Bookmarks** button, and the full list of previously saved bookmarks is shown.

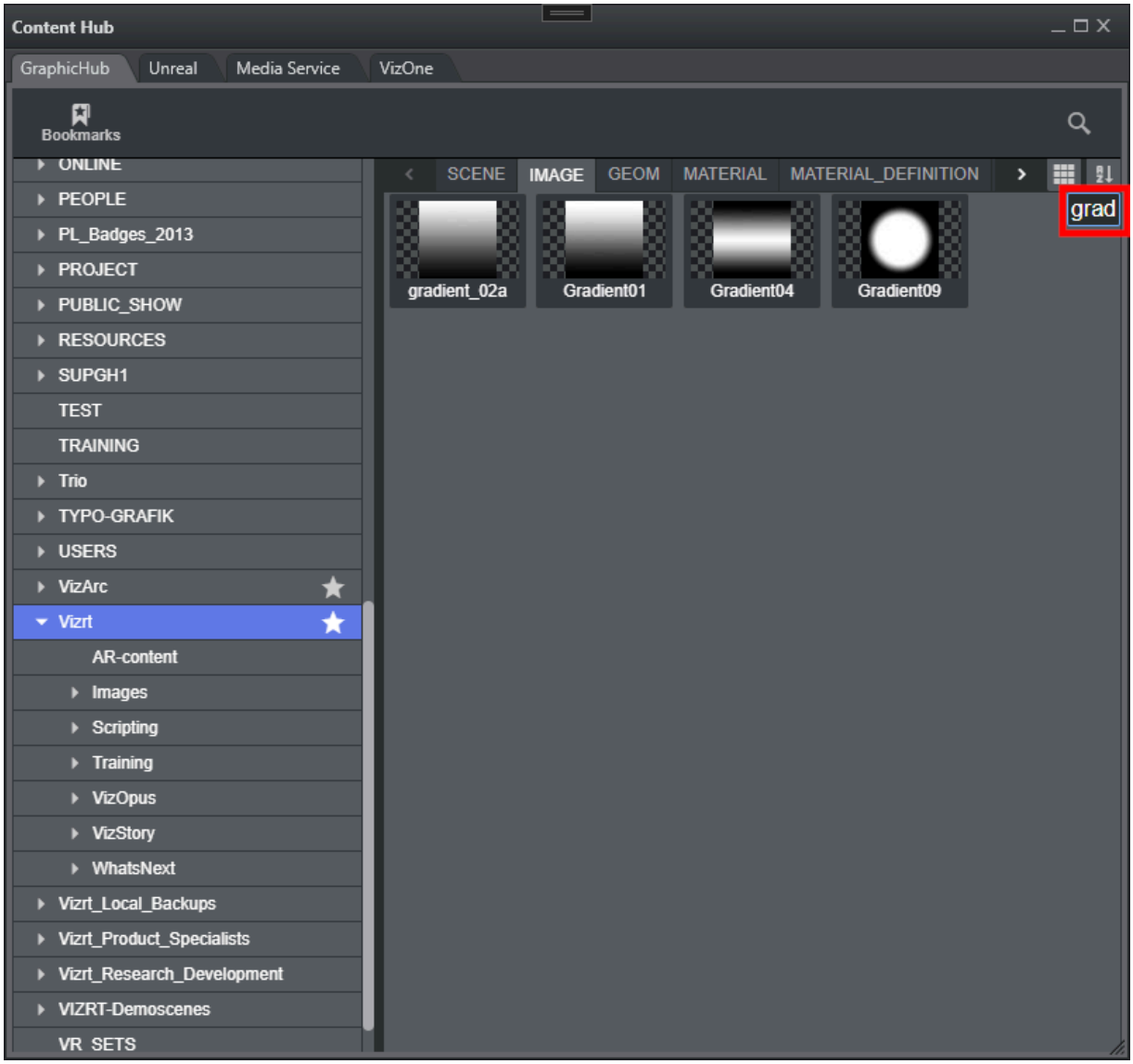
To delete a bookmark, right-click the bookmarked folder and then click on the **Remove bookmark** option.

Text Filtering

You can filter all the Assets whose filename contains a desired string by typing any alphanumeric character.

This works both when navigating the Folders and on the search results.

Copyright © 2025 Vizrt

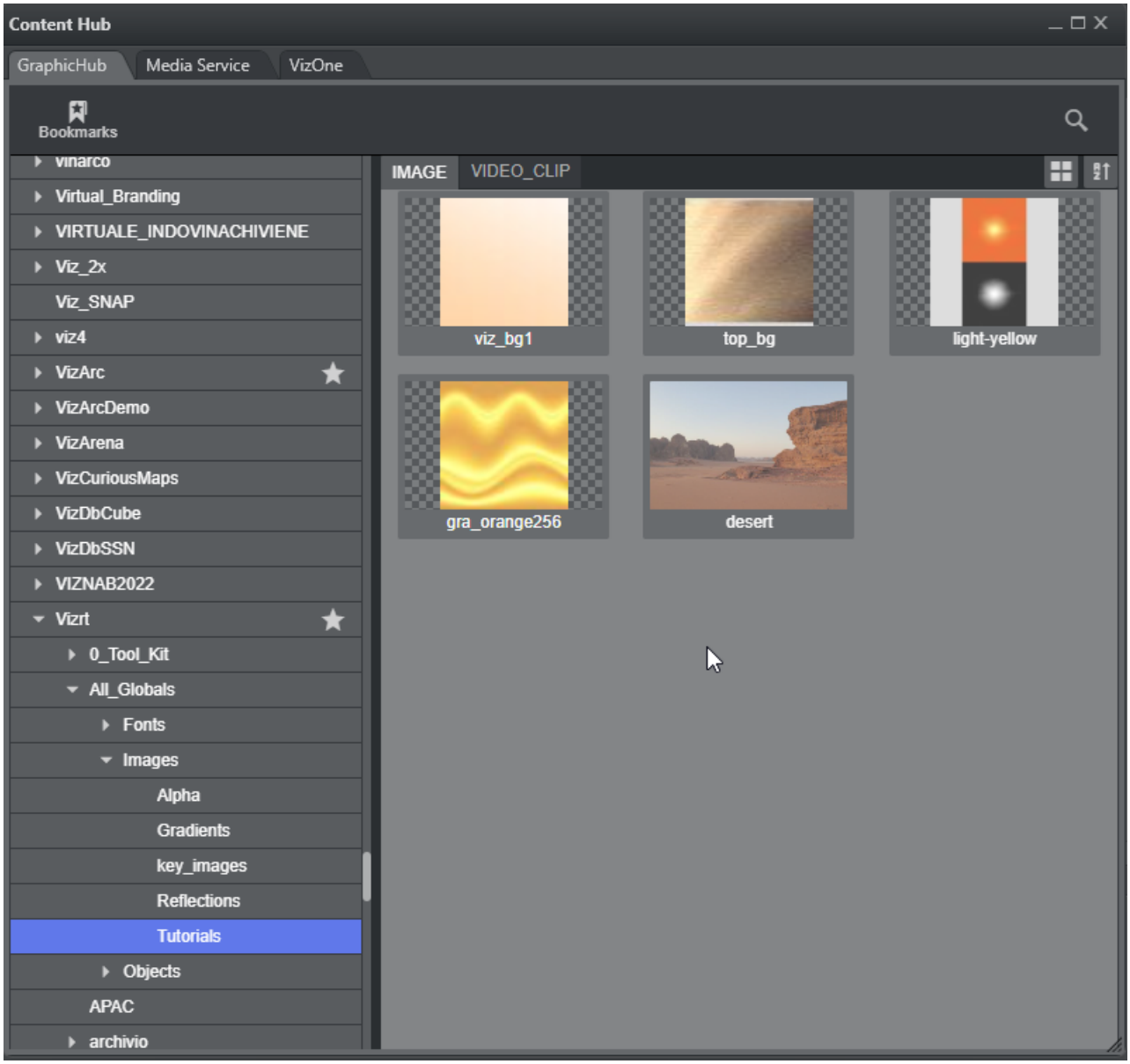


Import Images into the Graphic Hub

> Images can only be imported in the **IMAGE** tab. If there is no IMAGE tab available, it means that the asset type for the specific media cannot be an image. Please choose another one.

To import an image, it is necessary to select a folder and then drag one or more images from the user file-system to the right panel. An overlay appears as shown in picture.

Copyright © 2025 Vizrt



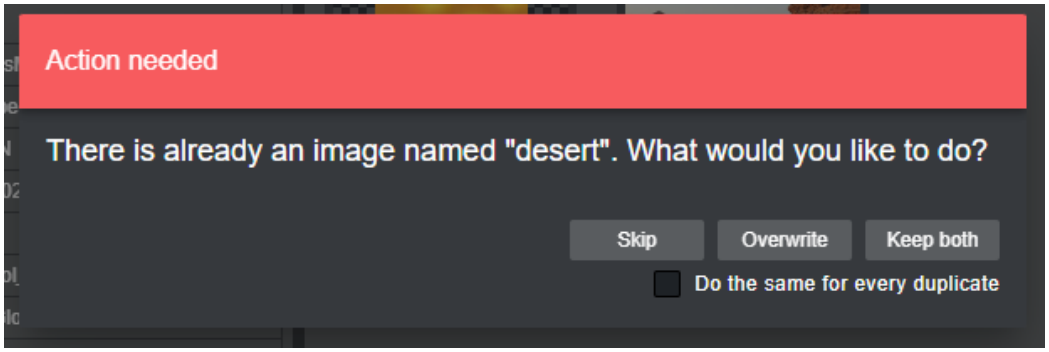
Once the image(s) is dropped it is automatically uploaded.

The accepted extensions are the followings:

- *image/jpeg*
- *image/png*
- *image/bmp*
- *image/tiff*
- *image/gif*
- *image/tga*

If there are one or more conflicts between filenames, a prompt appears asking the user what action you would like to take.

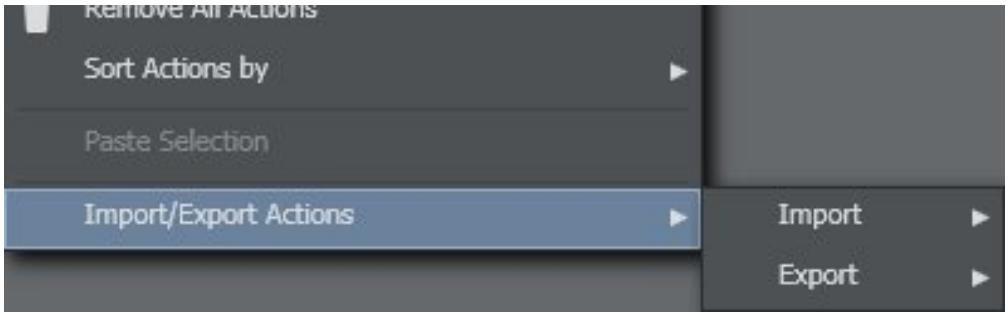
Copyright © 2025 Vizrt



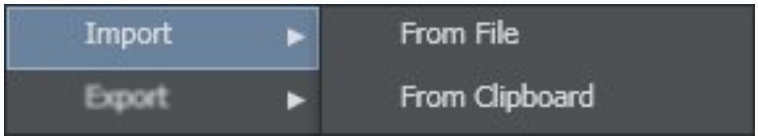
Copyright © 2025 Vizrt

5.11 Import and Export

By right-clicking on an empty space in the main area is also possible to import and export.

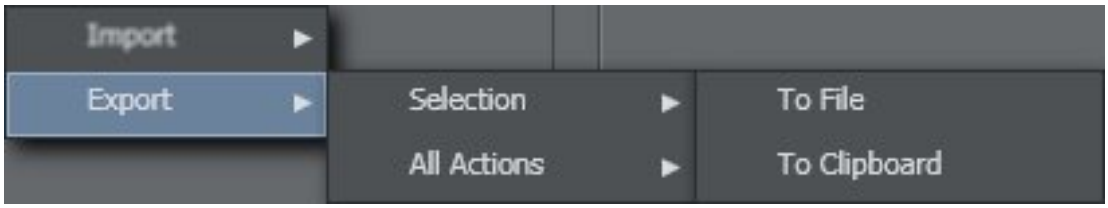


5.11.1 Import Actions



In Viz Arc, you can import action boxes directly into your active project. There are two ways to import boxes: internally (**From Clipboard**) or from an external file (creates a text file containing a *.json* structure). The boxes are imported directly into the same tab as they were located if such a tab exists, or imported into your active tab area.

5.11.2 Export Actions



You can export selected action boxes or all boxes in the opened project. You can choose to export boxes internally (**To Clipboard**) or to an external file (creates a text file containing a *.json* structure).

Copyright © 2025 Vizrt

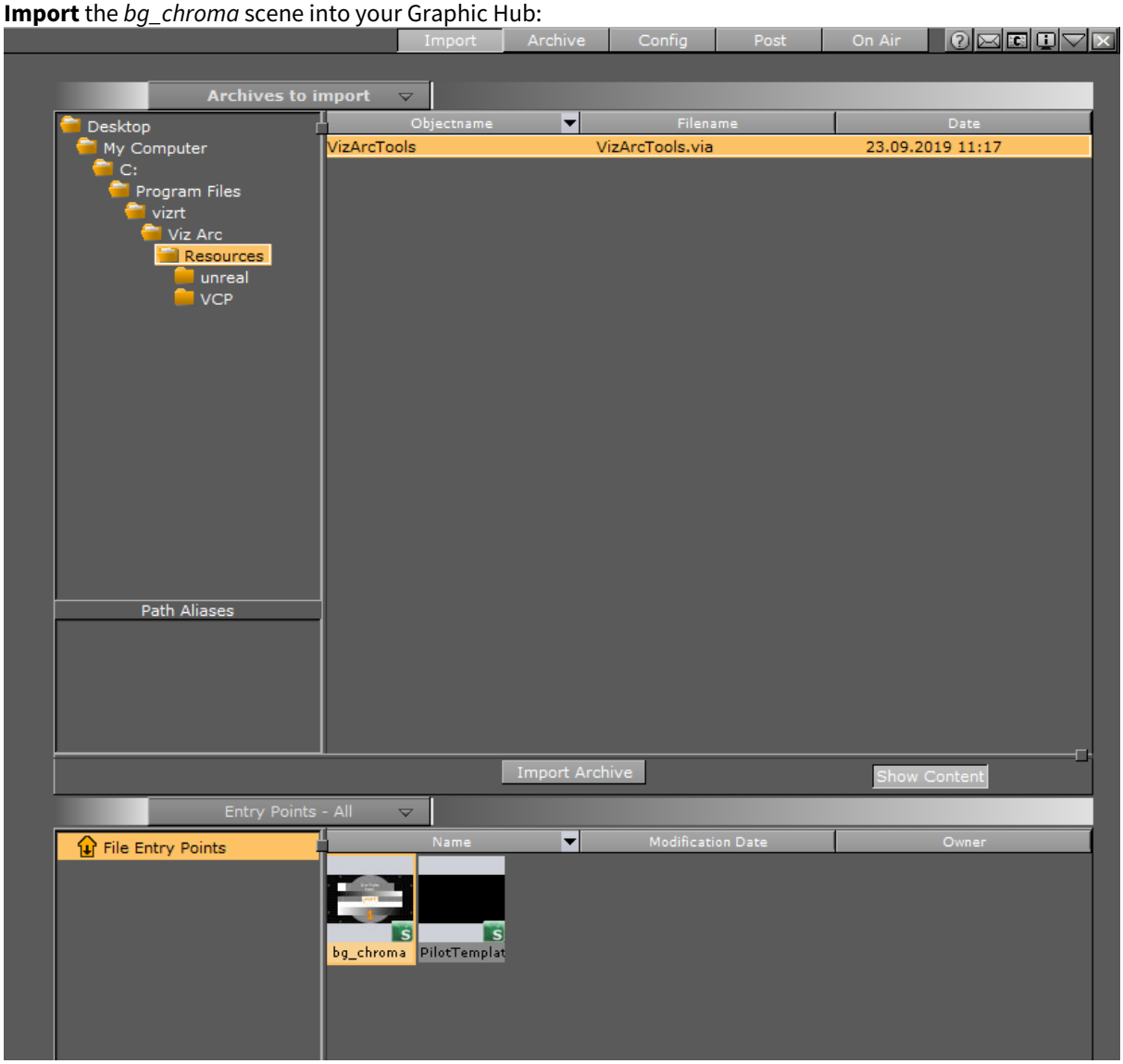
5.12 Multizone Chroma Keying

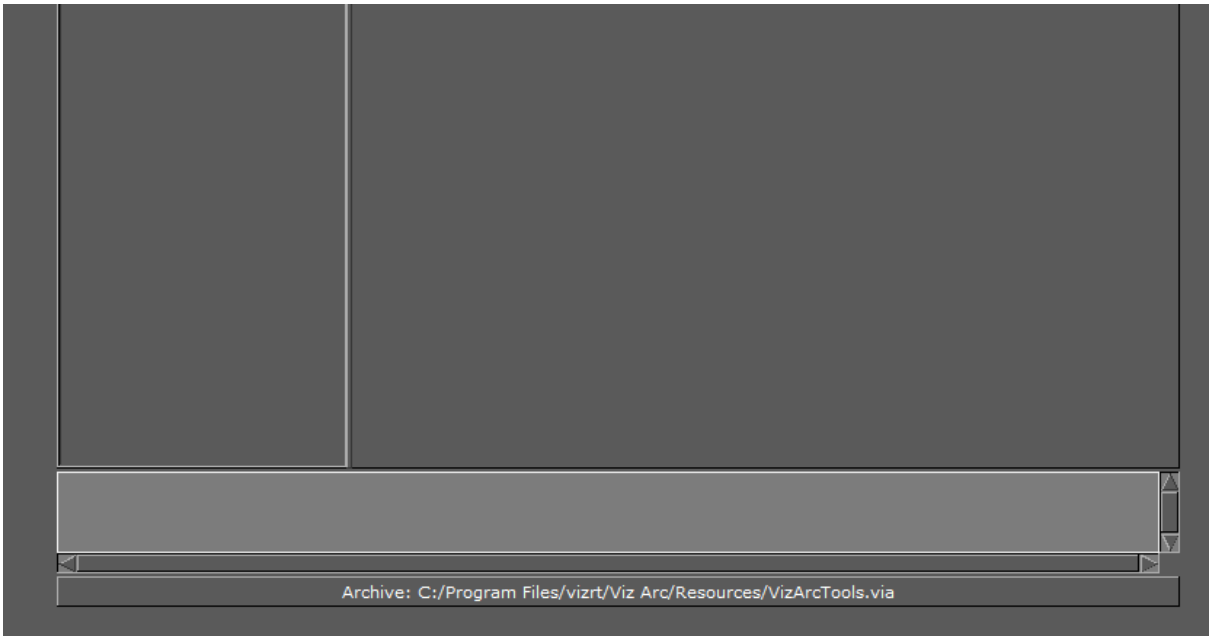
Use multizone chroma keying to define multiple chroma keying areas (or fullscreen) with different keying settings. Typical use-cases are areas with different colors or shades, such as horse racing tracks that have multiple shades of green. Multizone keying allows you to define specific zones of your tracked area where a specific key should be applied.

5.12.1 Configuring the Multizone Chroma Keyer

In order to make the multizone keying action work from Viz Arc you'll need to set-up a couple of things beforehand. The way the Multizone Chroma Keyer works in Viz Engine requires the Keyer to run in the back layer and the video input to be configured properly in mzone keying. Viz Arc ships with a template-based scene that you need to import into your Graphic Hub.

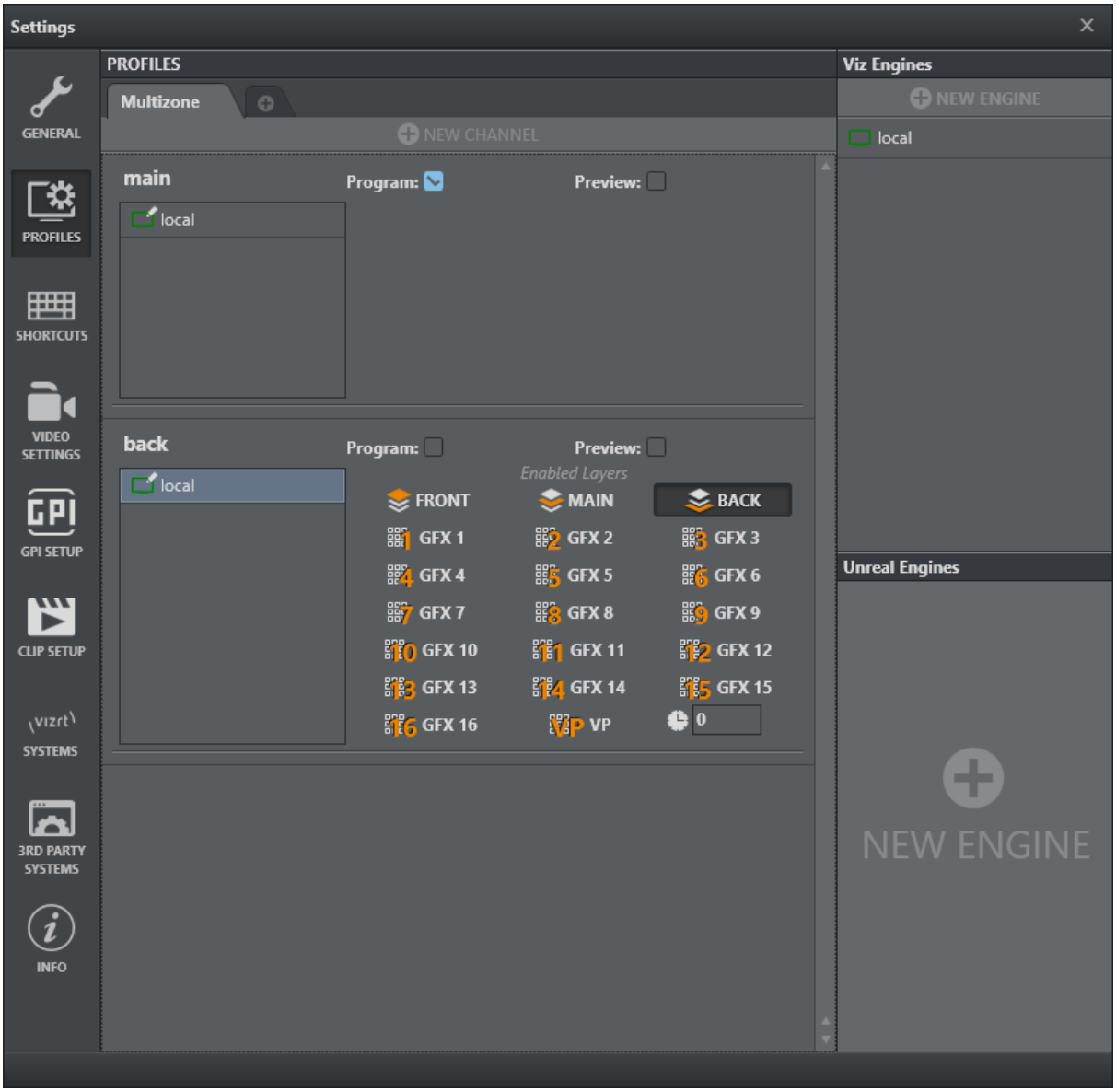
1. Locate the archive in the Viz Arc program-data folder (typically *C:\ProgramData\vizrt\VizArc*) under *Resources\MZChroma\VizArcTools.via*. The via file contains a scene named *bg_chroma*.
2. **Import** the *bg_chroma* scene into your Graphic Hub:





Since the scene is stored in the back layer of Viz Engine and all multizone controls are located there, you'll also need to configure a dedicated channel for the back layer in Viz Arc:

Copyright © 2025 Vizrt



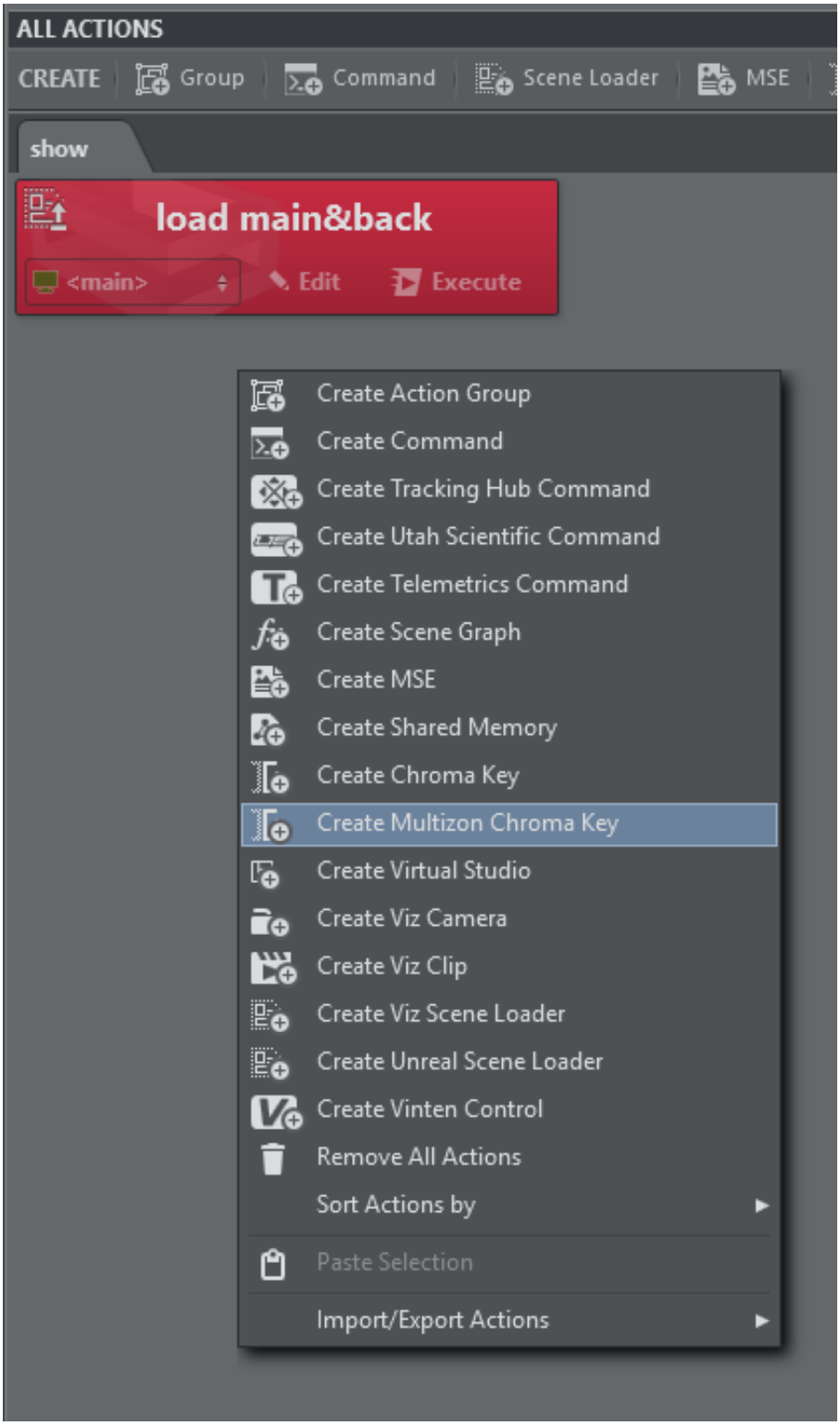
3. Add a channel to your Engine, uncheck the **MAIN** layer and check the **BACK** layer only. Give the channel a meaningful name like "back". The other channel (in this case **main**) is used for the actual AR/VR graphics, where only the **MAIN** layer is checked (default).
4. The next step is to create a scene loader by right clicking the action canvas and selecting **Create Viz Scene Loader** . This loads the recently imported scene into the back layer. You can also load your main scene as in the same loader.

Copyright © 2025 Vizrt



5. Once the background chroma scene has been loaded into the Viz Engine and a valid video signal has been connected, you can start adding a multizone chroma action to the Viz Arc action canvas. To do so, right click on the action canvas and select **Create Multizone Chroma Key** .

Copyright © 2025 Vizrt

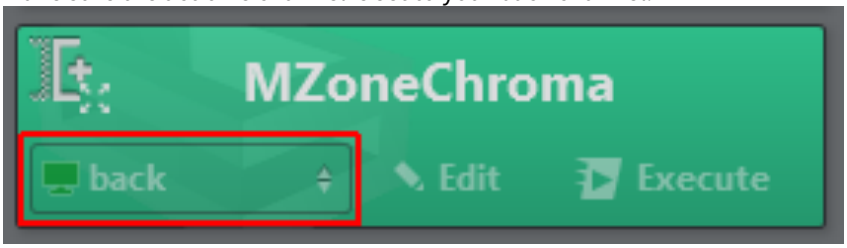


Once the action has been created you can fill in a name for a new full screen multizone, for example **fsGreen** .

Copyright © 2025 Vizrt

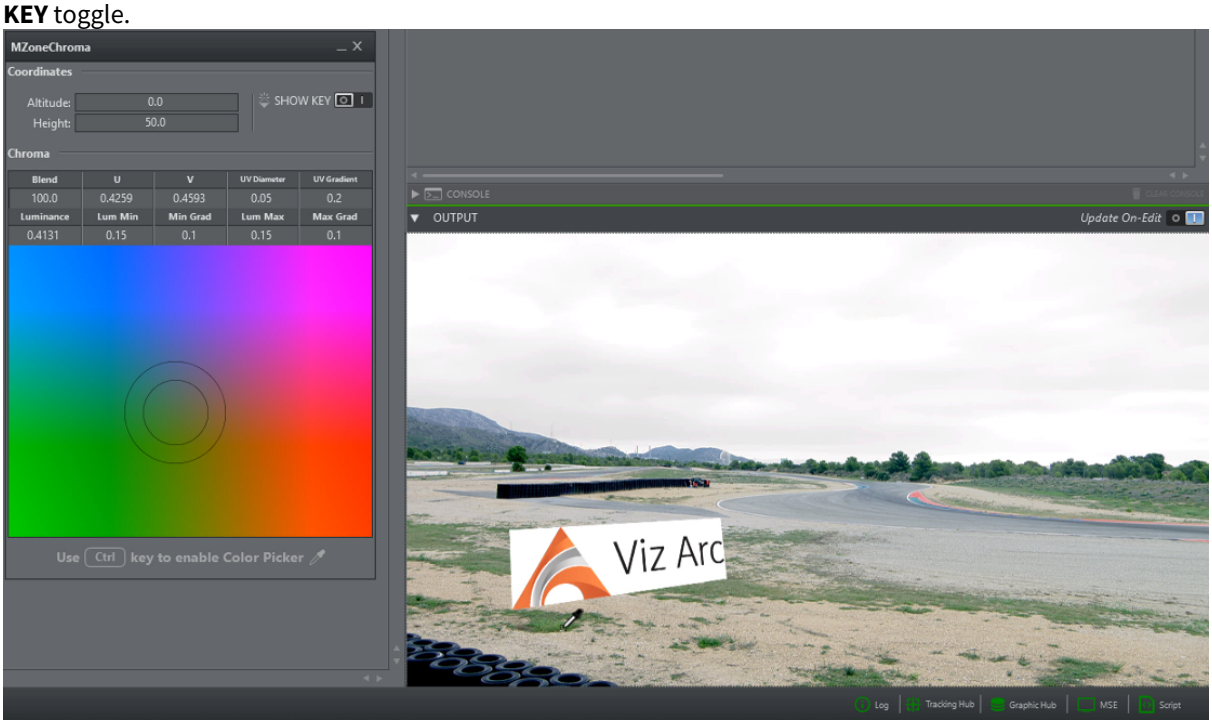


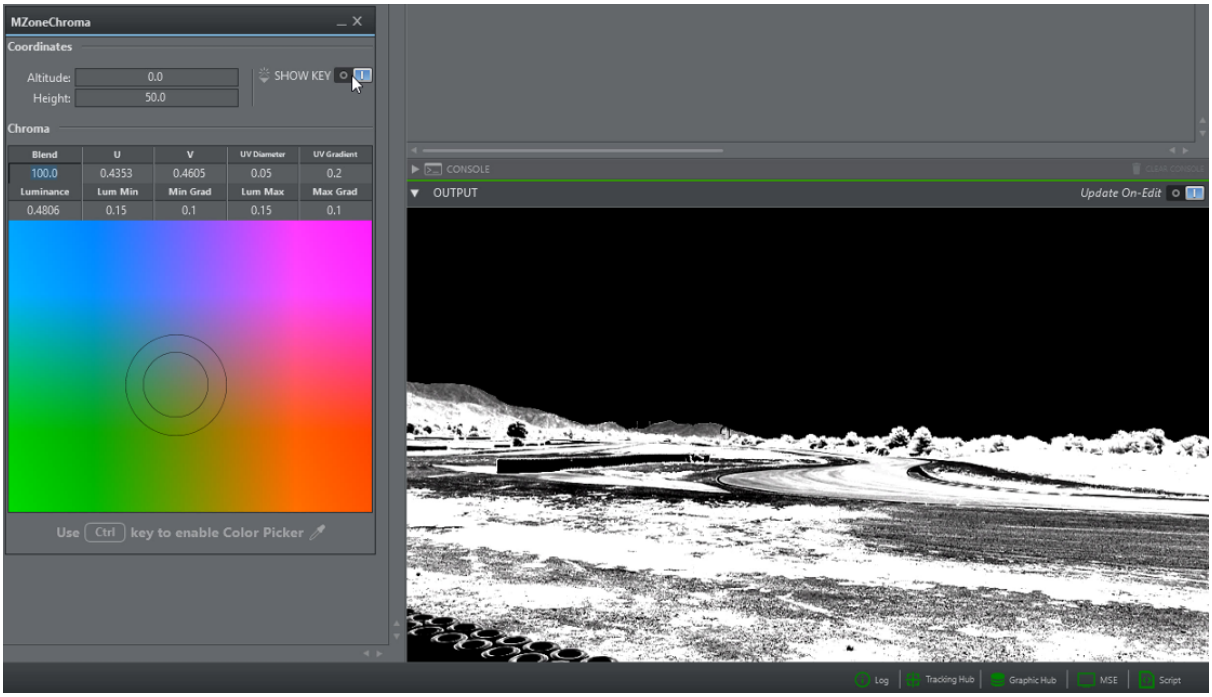
Make sure the action's channel is set to your back channel.



Copyright © 2025 Vizrt

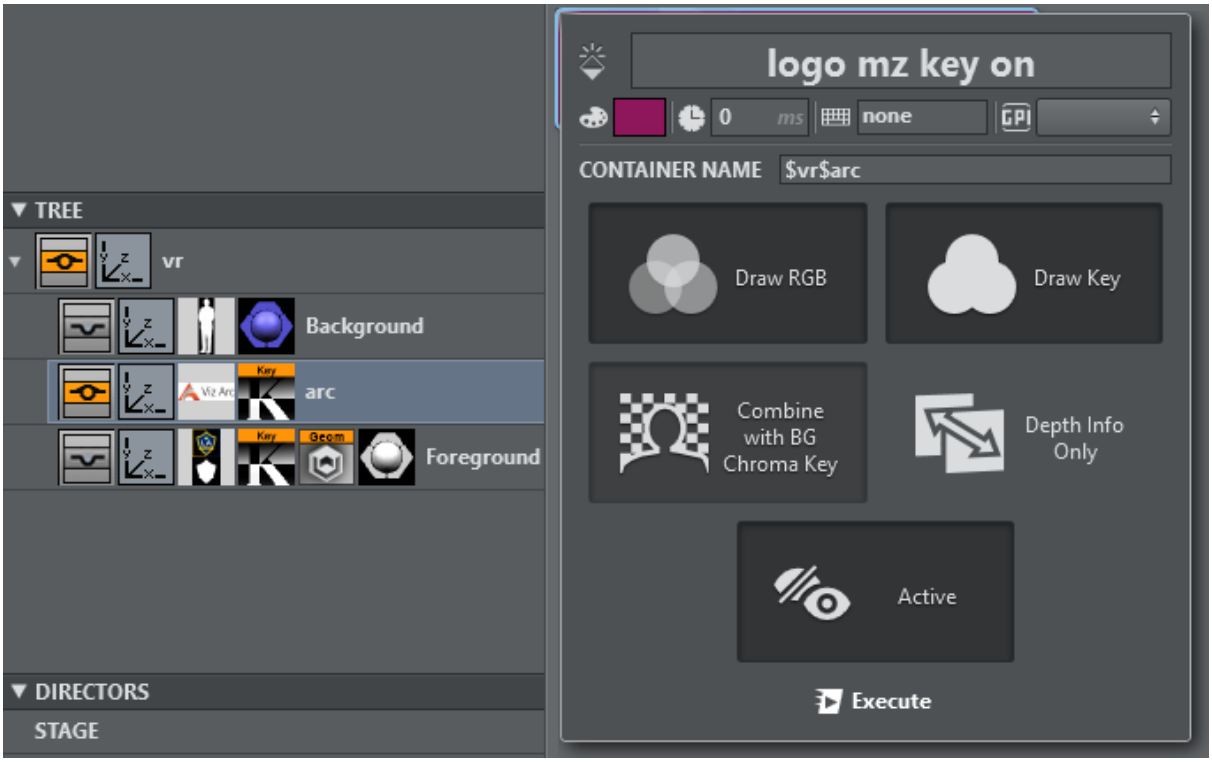
You can now start using the picker on your output directly or you can adjust the key manually. For picking, it's recommended to use the **Window** mode to make sure the window remains open after picking. Hold down **CTRL** and click on the preview output. In order to see the actual key on the output, press the **SHOW KEY** toggle.





Once you have adjusted the key, you can test it with an element in your main scene. To do this, create a **Key Action** by dragging the key plugin to the action canvas. In the keying action enable **Combine with BG Chroma Key**.

Copyright © 2025 Vizrt

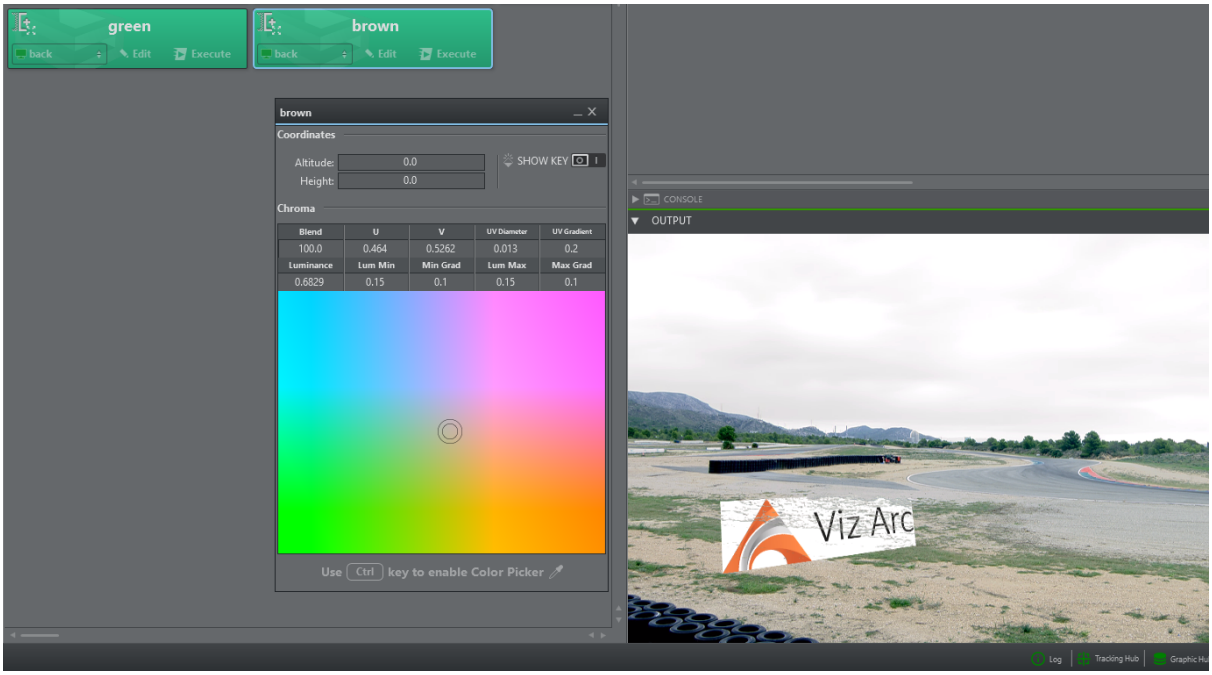


You'll now see the respective Viz object keyed on the green areas in the output.



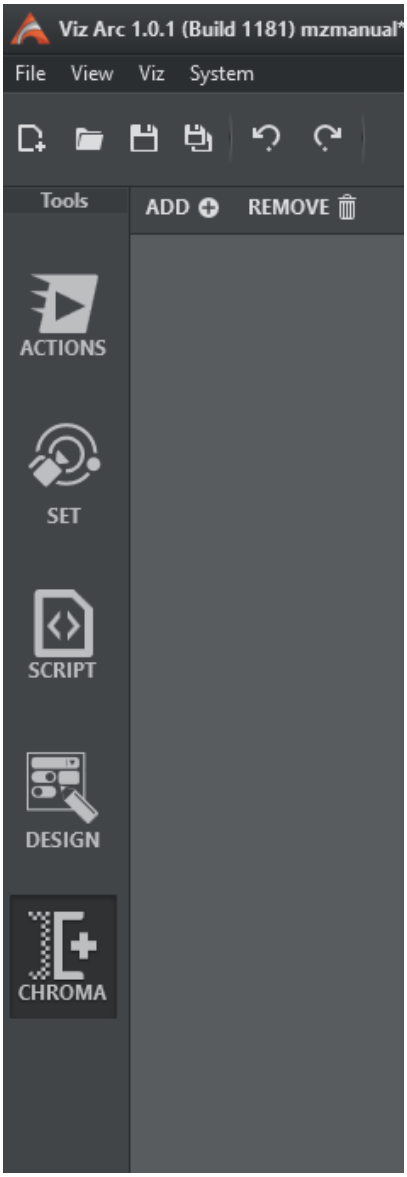
You can now add more Multizone Chroma Actions to key more colors, for example a key for the color brown.

Copyright © 2025 Vizrt

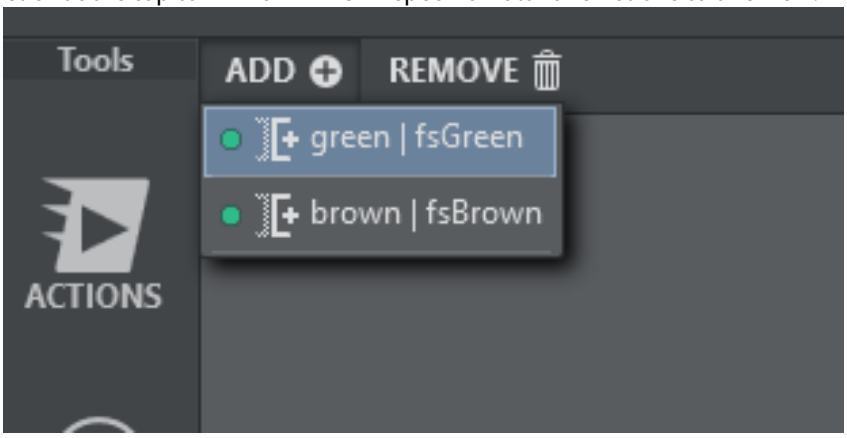


To make On Air operation more efficient, you can use the dedicated **CHROMA** view for quick operation. This button appears as soon as a multizone chroma action is present.

Copyright © 2025 Vizrt

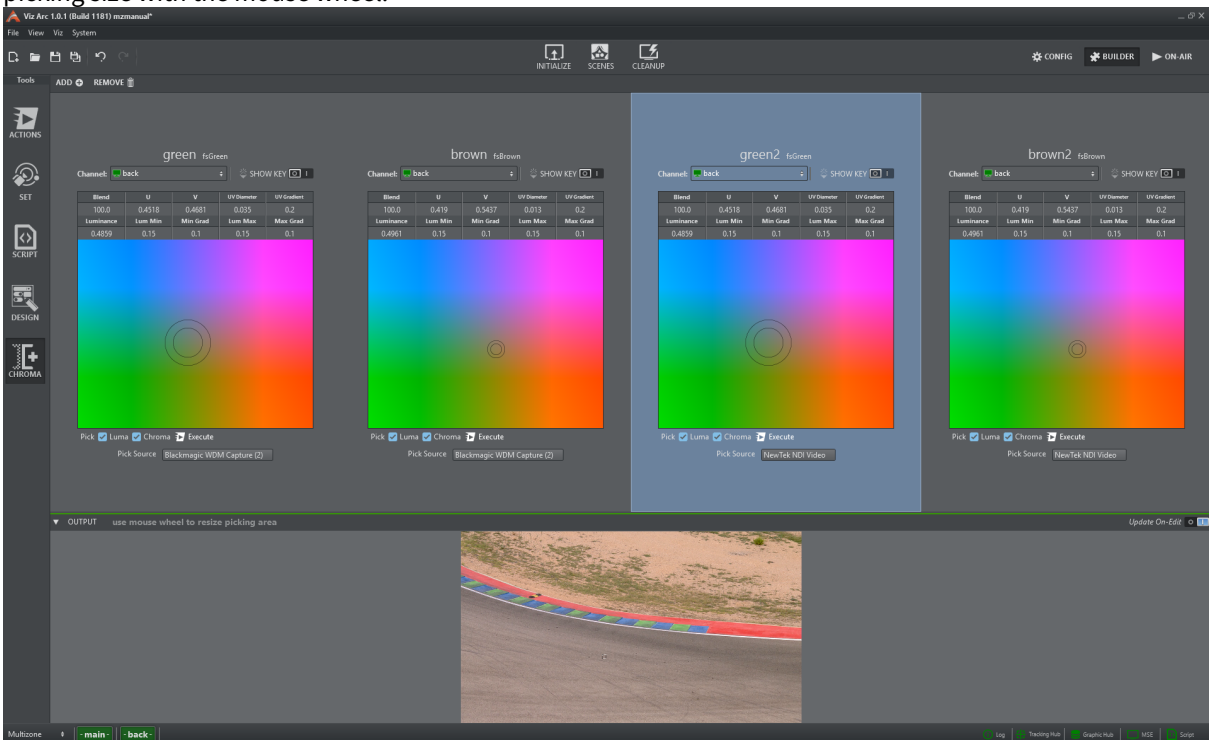


Click at the top to **ADD** or **REMOVE** specific Multizone Actions to this view.

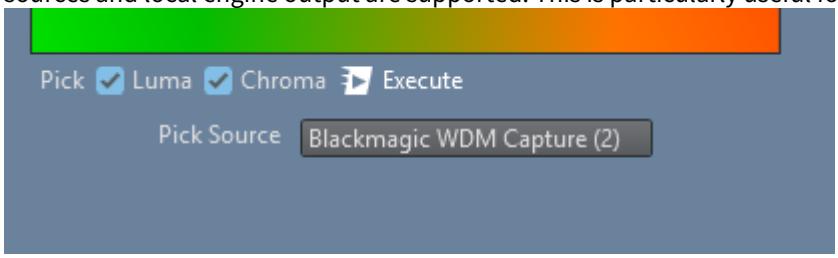


Copyright © 2025 Vizrt

Once you have added all relevant keys for your show, you can easily navigate through the keys using the arrow keys on your keyboard. You can directly pick in the video window where you can also adjust the picking size with the mouse wheel.



In **Pick Source** , you can select the signal that is to be shown when the chroma key is selected. Both DirectShow sources and local engine output are supported. This is particularly useful for multi-camera set ups.



You can also specify what is to be evaluated during picking.

- Check **Luma** if you just want to pick the luminance (for example, if you want to react only to changes in brightness shading).
- Check **Chroma** if you want to pick the color too.
- Or check both if you want to change both of them.

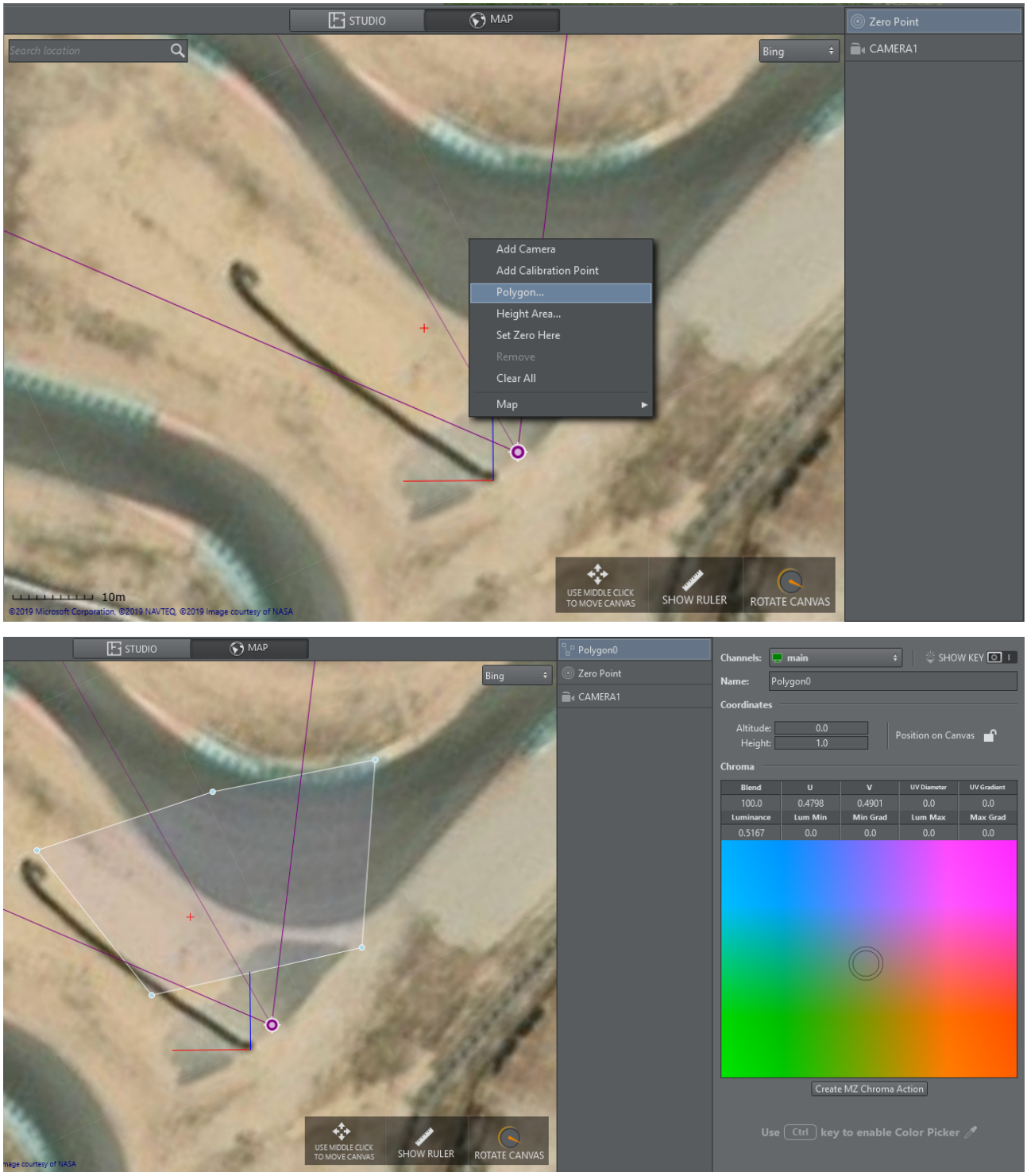
The keying values are sent to the engines directly as long **Update On-Edit** is enabled.

You can pick by clicking the **left cursor button** on the output window. The size of the picking area can be adjusted with the **mouse wheel**. The **space bar** freezes or unfreezes the output window.

5.12.2 Working with Polygons and the Map

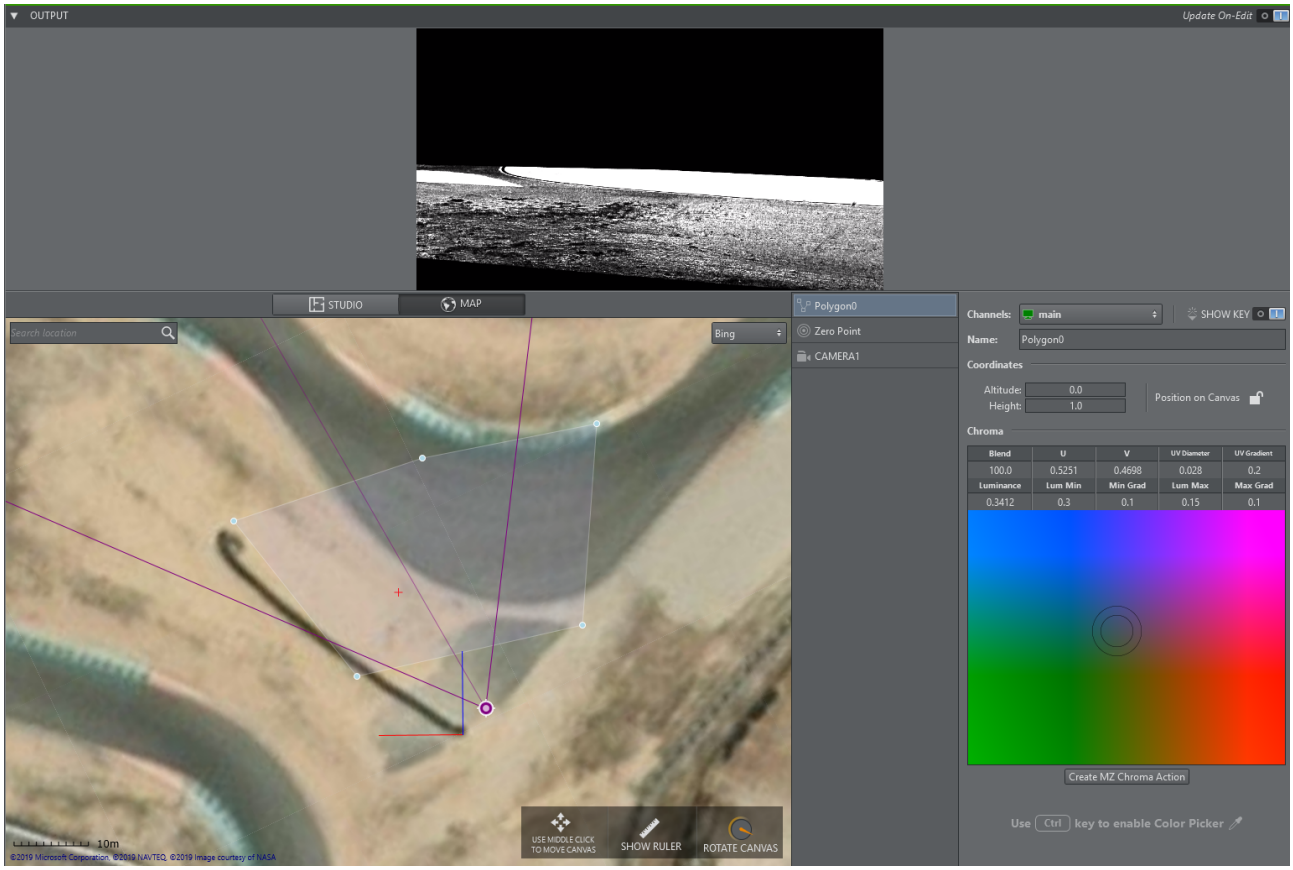
If you want to restrict your key to a certain area or shape, you can use polygon shapes that can be defined in the map view. Go to **MAP**, right click and select **Polygon...**

Copyright © 2025 Vizrt



Click on the map for each vertex and finish the drawing by pressing the **ESC** key. You'll see a multizone chroma control on the right hand side. If you now click on **Create MZ Chroma Action**, a corresponding action is inserted on the action canvas.

Copyright © 2025 Vizrt



Viz Arc creates a polygon in the background scene that is tracked with the tracked camera. The area that is keyed on remains fixed in the same place. Note that the polygon is flat and a flat floor is therefore assumed.

Copyright © 2025 Vizrt

5.12.3 Multizone Chroma Key Action in Detail



Copyright © 2025 Vizrt

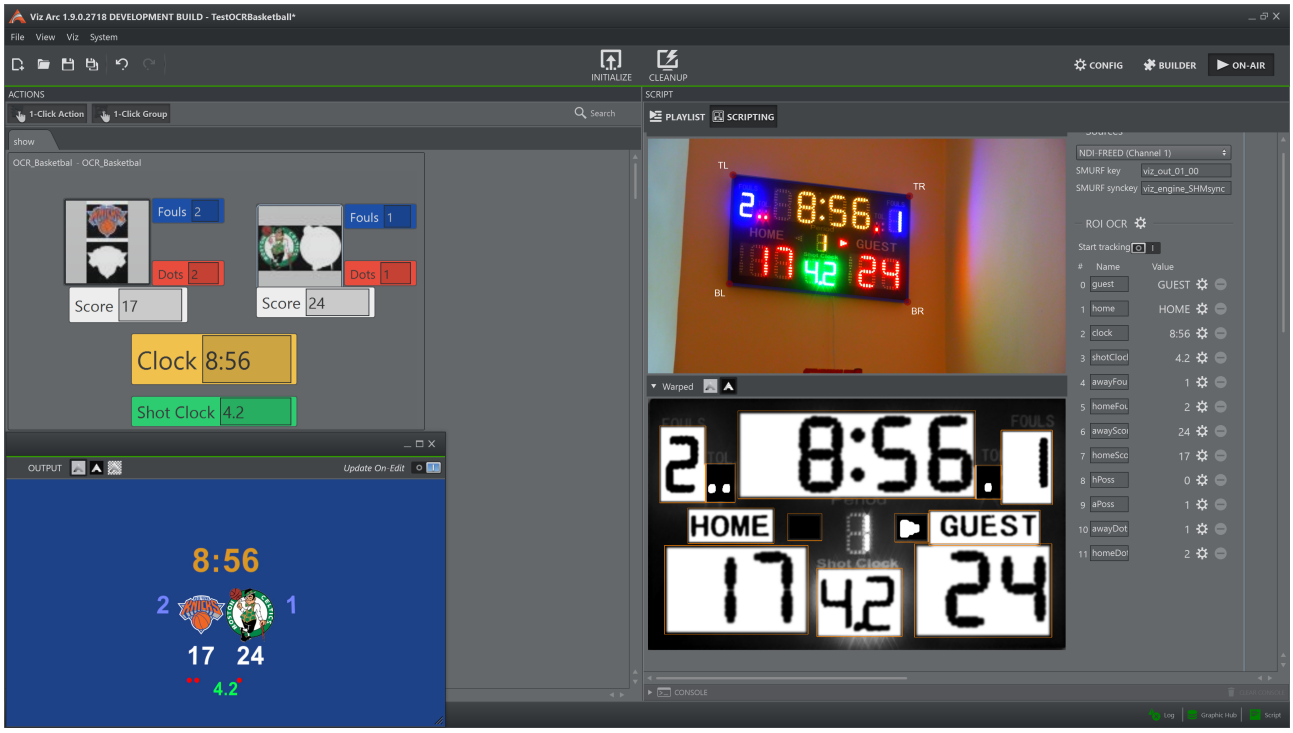
Blend	Alpha of key signal, 100 is full opacity, 0 is completely transparent.
U	U value for color keying.
V	V value for color keying.
UV Diameter	All colors between the reference UV and this diameter are removed.
UV Gradient	Gradient for fade out at diameter border (0=sharp, 1=soft).
Luminance	Luminance is defined as 0.0 for dark and 1.0 for bright.
Lum Min	Lower luminance limit for keyed colors.
Min Grad	Gradient for lower luminance limit (0=sharp, 1=soft).
Lum Max	Upper luminance limit for keyed colors.
Max Grad	Gradient for upper luminance limit (0=sharp, 1=soft).

Note: Lum Min and **Lum Max** are not absolute values, but are relative to the **Luminance** (e.g. when **Luminance** is 0.5, **Lum Min** is 0.1 and **Lum Max** is 0.2, the Keyer works in the range 0.4 to 0.7).

Copyright © 2025 Vizrt

5.13 OCR

The OCR (Optical Character Recognition) widget enables to recognize any characters or numbers from a video source and store them in the DataMap.

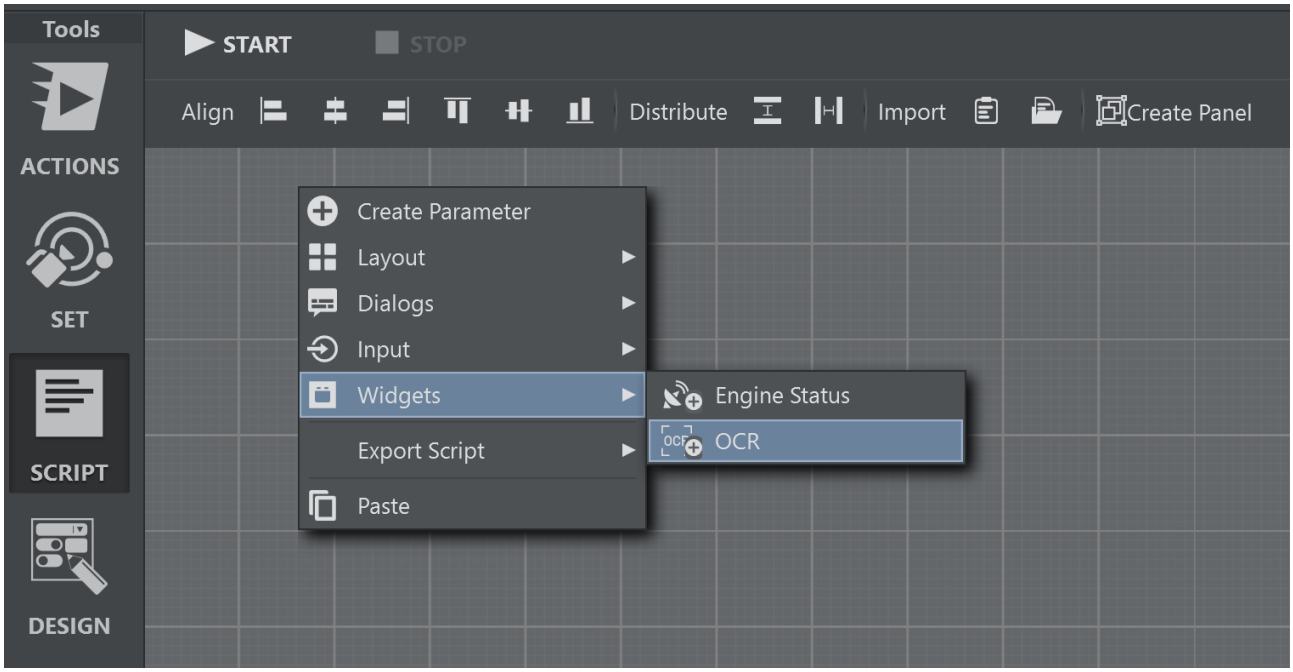


- To Add an OCR Widget
  - Sources
  - Warp
  - Global Settings
  - ROI (Region Of Interest)
    - Add a New ROI OCR
    - Properties
    - ROI Settings
    - Example
    - Templates and DataMap
  - Tracking

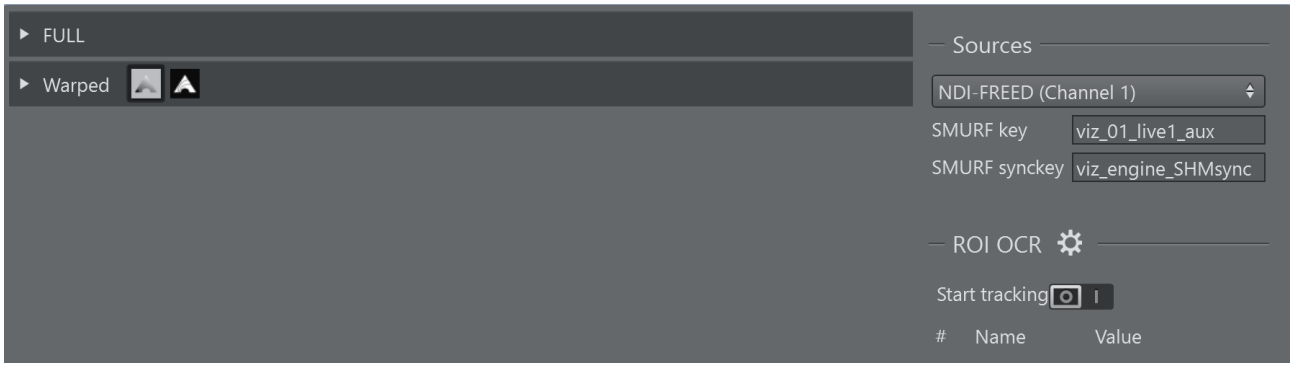
5.13.1 To Add an OCR Widget

Go to the main script (or any template script), right click on the UI editor and select **Widgets > OCR**.

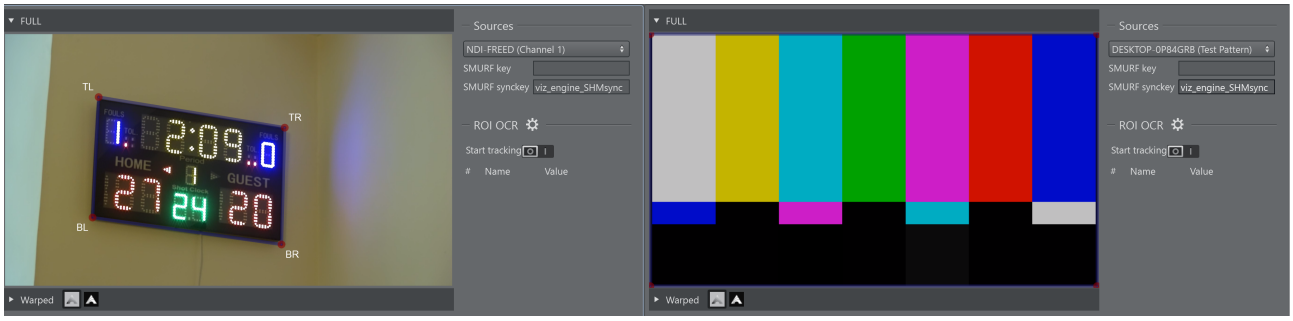
Copyright © 2025 Vizrt



Resize and position the widget. Start the script.



You can add multiple widgets.



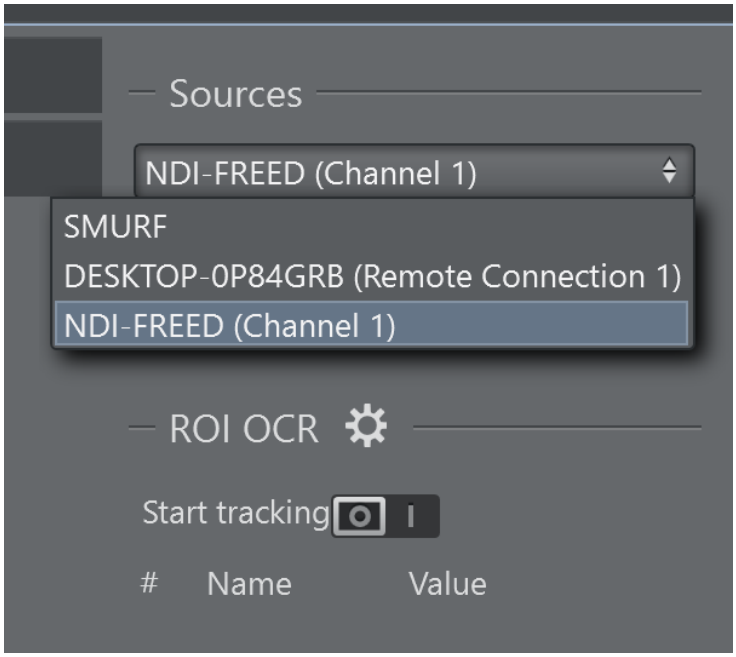
Sources

Select a **Source** in the drop down menu. The widget supports two types of sources:

- **SMURF:** Read from a shared data stream identified by given key. This needs both the key and the sync key to work.

Copyright © 2025 Vizrt

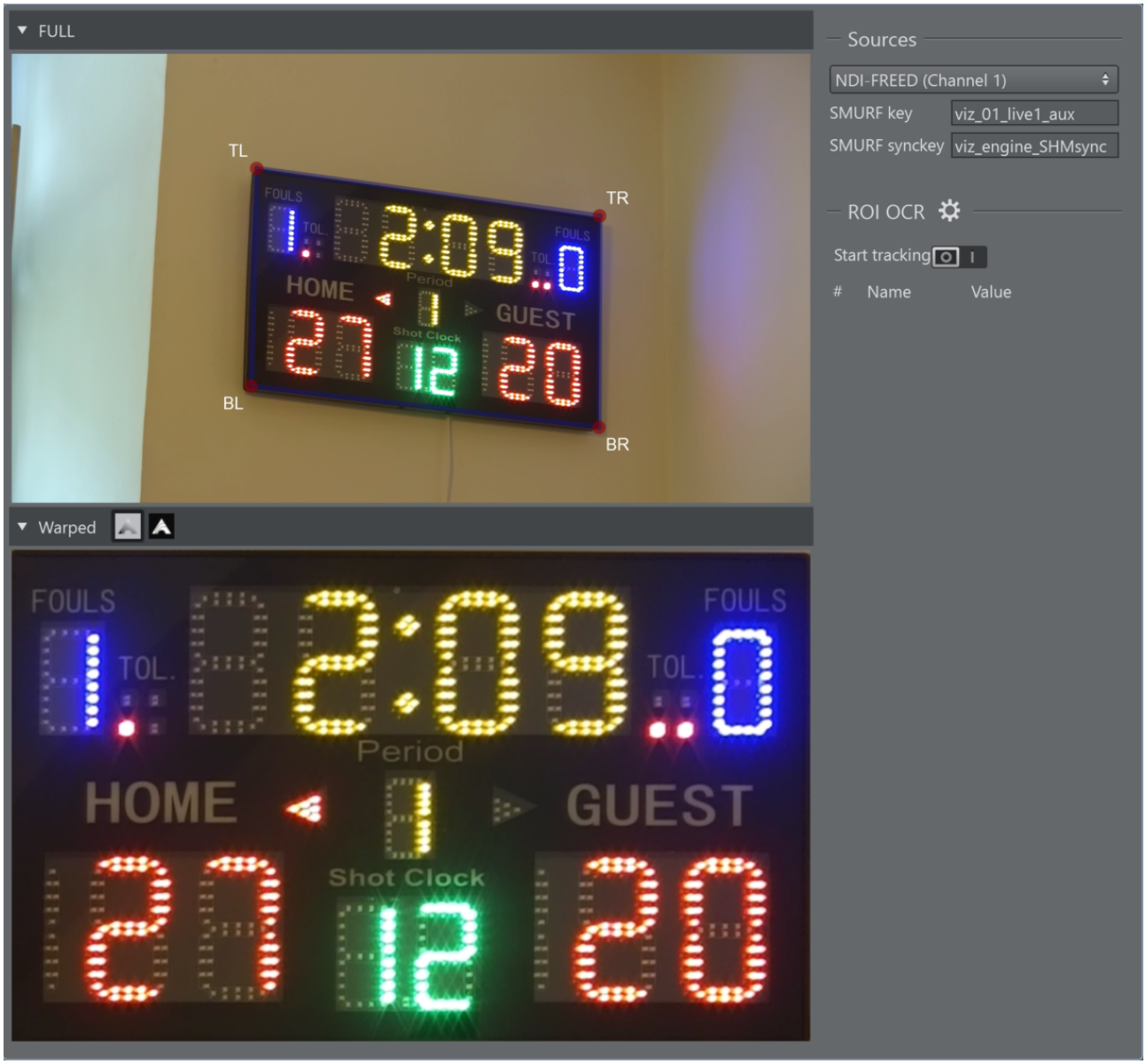
- **NDI:** Each NDI stream found is listed in the drop down.



Warp

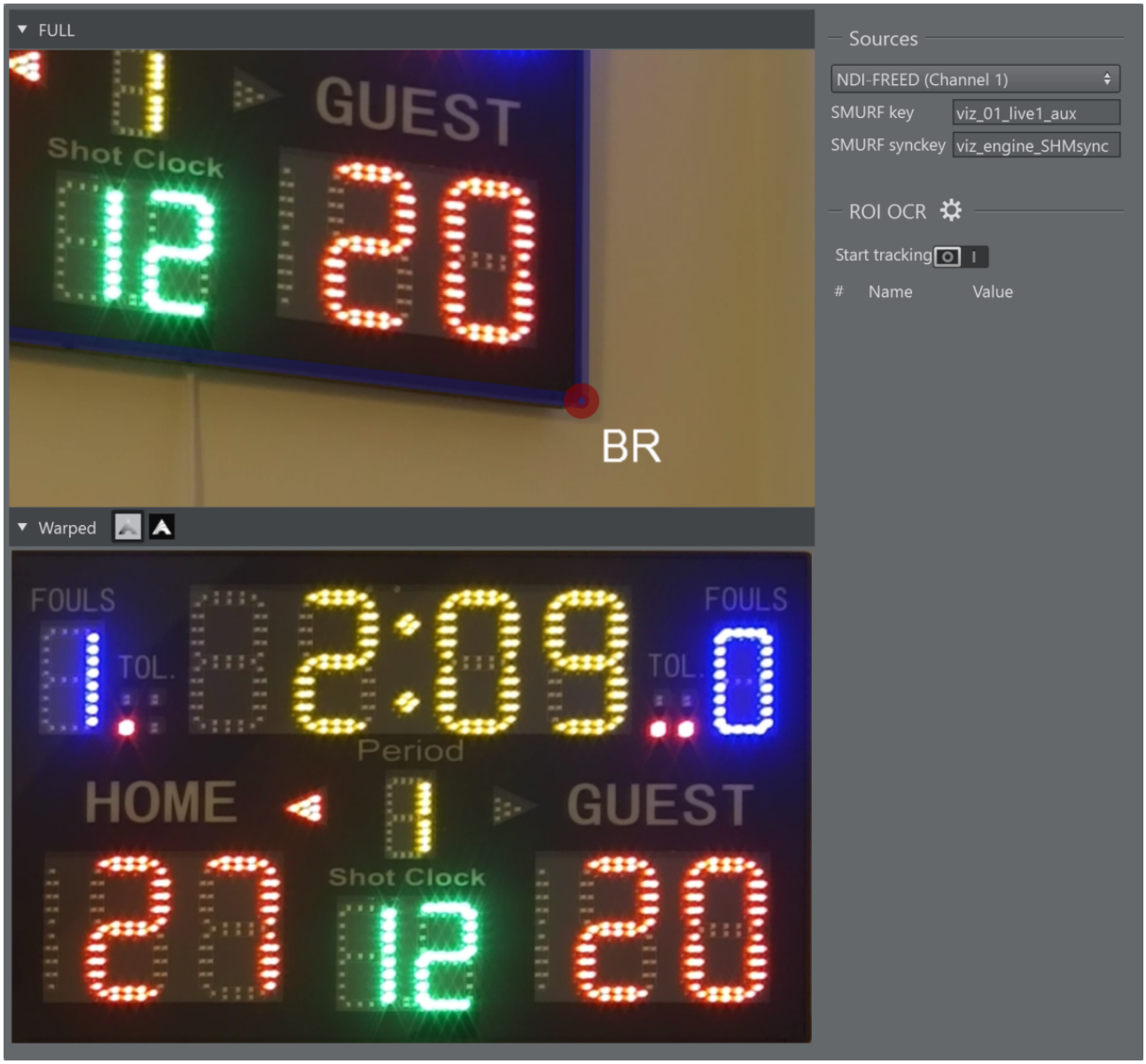
Herdly the camera is not perpendicular to the target (ex. scoreboard), so the image needs to be warped. This step helps the OCR algorithm and improves the results.

Wrap the quad around the target grabbing the red circles.



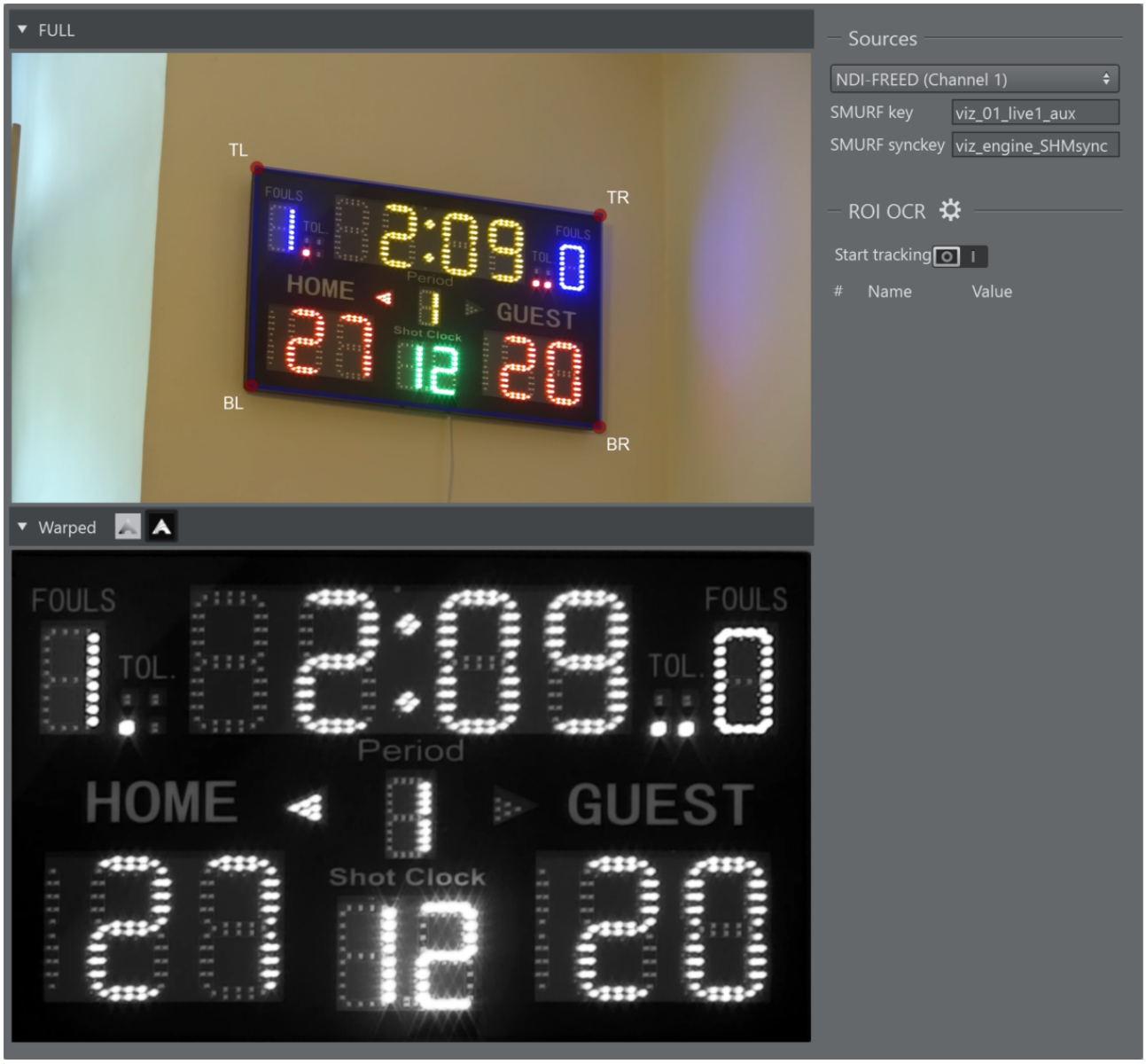
Zoom the image with the mouse wheel, helping to wrap the target more precisely.

Copyright © 2025 Vizrt

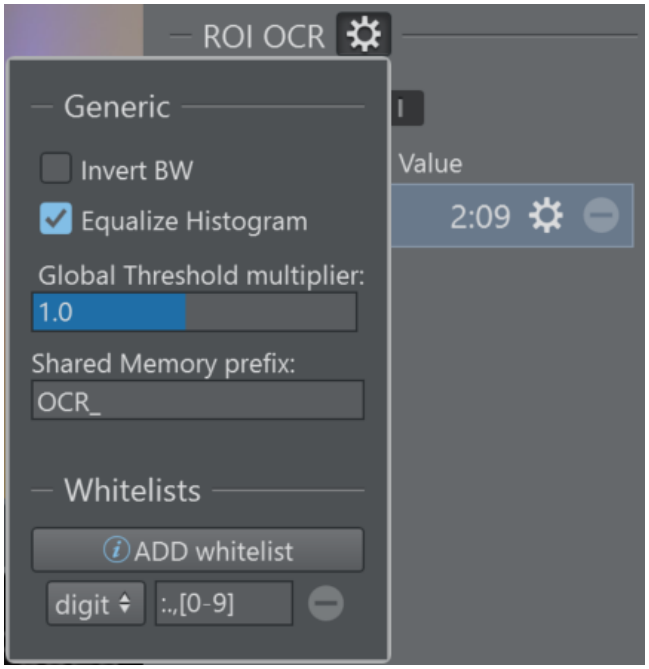


The OCR algorithm works with black and white images; to see the actual image used, you can click on the **Show Key** button next to the **Warped** expander.

Copyright © 2025 Vizrt


FULL
TL
TR
BL
BR
FOULS
TOL.
2:09
Period
HOME
GUEST
Shot Clock
Warped
Sources
NDI-FREED (Channel 1)
SMURF key
viz_01_live1_aux
SMURF synckey
viz_engine_SHMsync
ROI OCR
Start tracking
Name Value

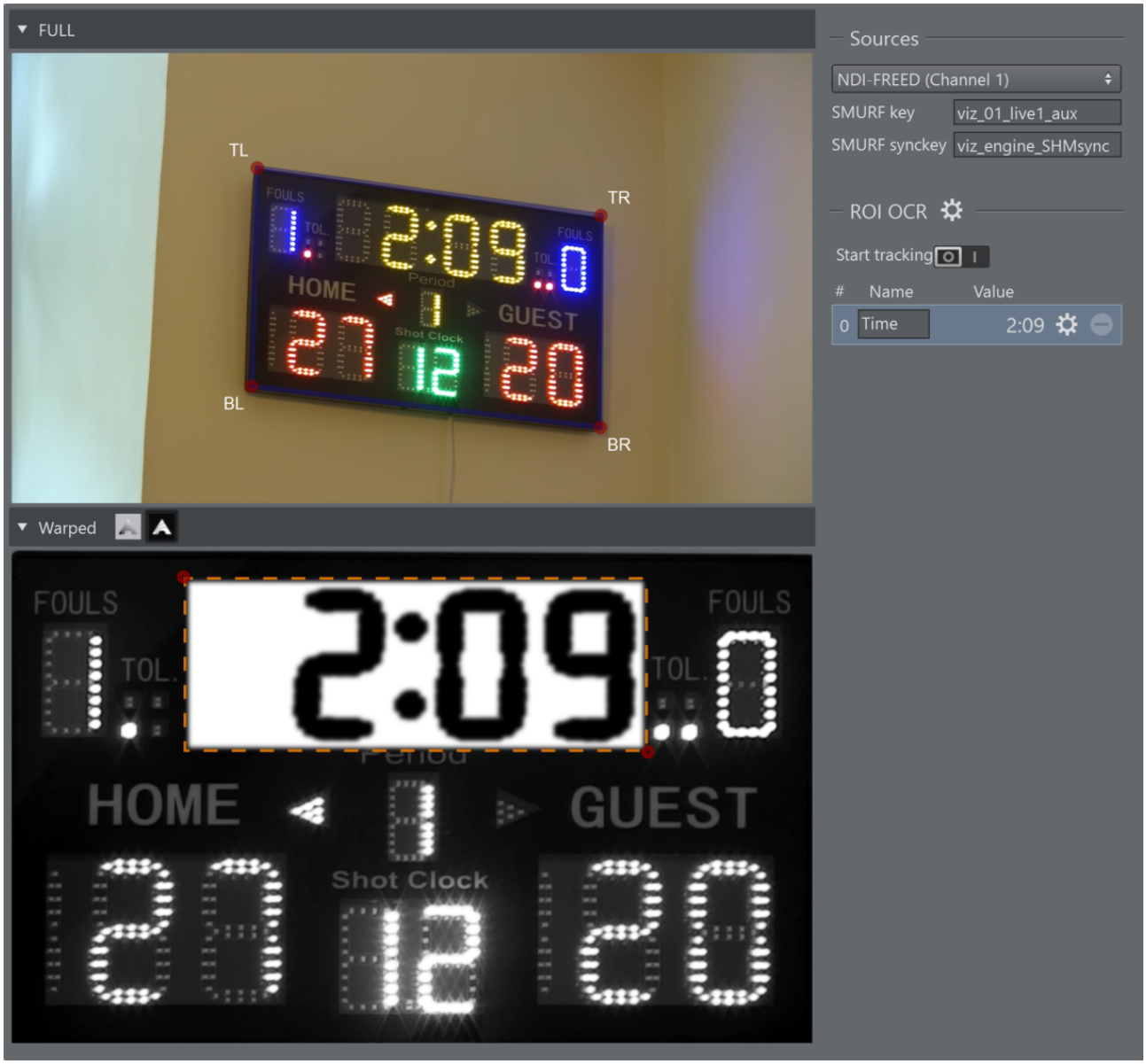
Global Settings



- **Invert BW:** The OCR algorithm works best with black chars on white background, if the result of binarization is inverted you can click on this checkbox to fix it.
- **Equalize Histogram:** Enhances the contrast and helps to be less sensible to light changes.
- **Global Threshold multiplier:** If there is a light change that leads the algorithm to mixed results, it is possible to use this slider to change all the threshold values of the OCR Rois.
- **Shared Memory prefix:** This string is prepend to the OCR Roi name and the result is used as a key in the share memory datamap.
- **Whitelists:** It is possible to suggest the allowed chars to improve the results, it doesn't support regular expressions but has some keywords:
    - *[a-z]* all lowercase letters.
    - *[A-Z]* all uppercase letters.
    - *[0-9]* all digits.
    - *[a-zA-Z]* all letters.
    - *[a-zA-Z0-9]* all letters and digits.

Copyright © 2025 Vizrt

ROI (Region Of Interest)



Add a New ROI OCR

- Click and drag the left mouse button in the Warped View to begin drawing the rectangle from its first corner.
- While holding the left mouse button, drag to the required size.
- Releasing the left mouse button.

It is possible to adjust the size of the ROI by grabbing the top left corner or the bottom right.

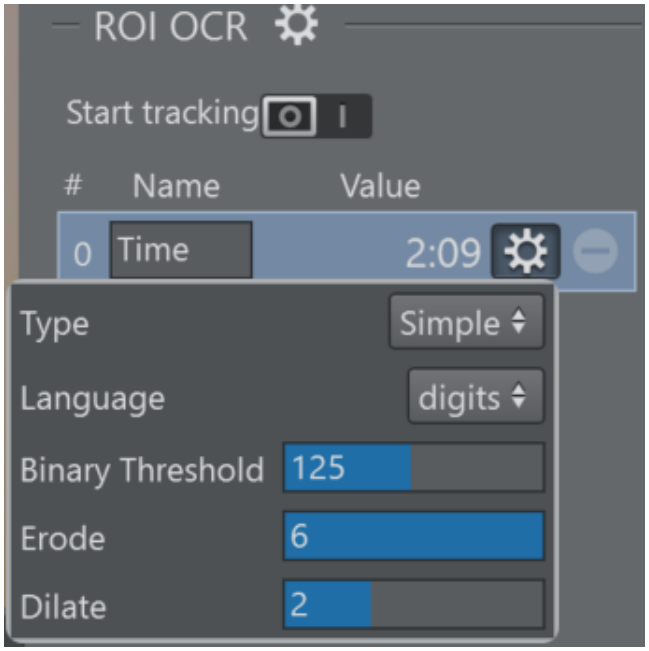
Properties

- **ID:** The ID is used as part of the DataMap key if the name field is empty.

Copyright © 2025 Vizrt

- **Name:** Second part of the DataMap key (the first is the prefix see Config paragraph above).
- **Value:** Result of the OCR algorithm that is sent to the DataMap.
- **Settings Button:** See below.
- **Remove Button:** To remove the ROI.

ROI Settings

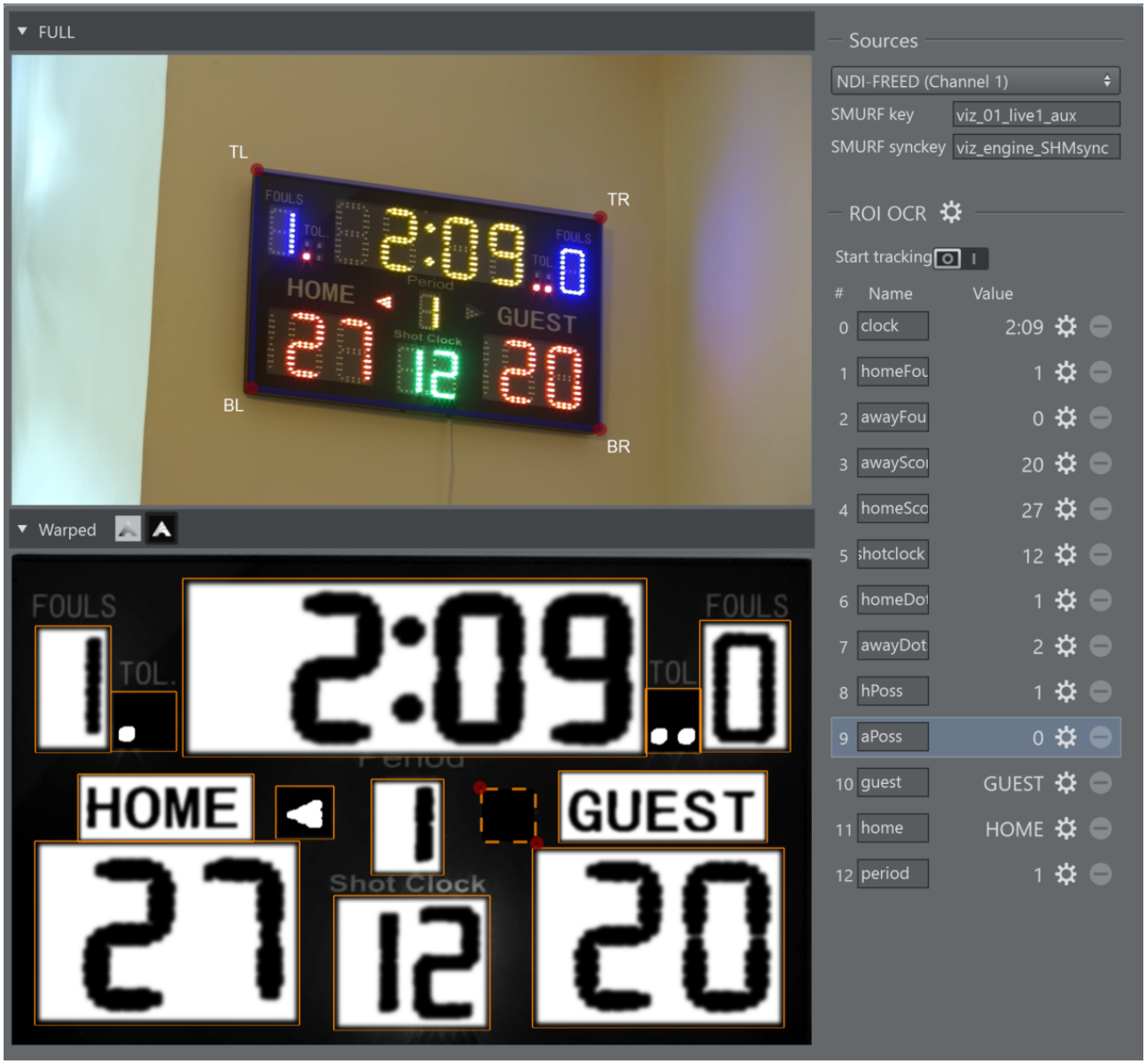


- **Type:**
  - Simple: Normal OCR algorithms.
  - Dots: Count the dots in the ROI.
- **Language:** Language selection to improve the results (even better if used in conjunction with the *whitelist*)
- **Binary threshold:** The OCR algorithm works on black and white images, therefore a threshold must be applied. In the **Warped** option, if you click on the **ShowKey** button you'll see a grayscale image where the pixel value range is between 0 and 255, so every pixel below the threshold is white or black if above.
- **Erode:** The bright areas of the image get thinner, whereas the dark zones gets bigger.
- **Dilate:** The dark areas of the image get thinner, whereas the bright zones gets bigger.

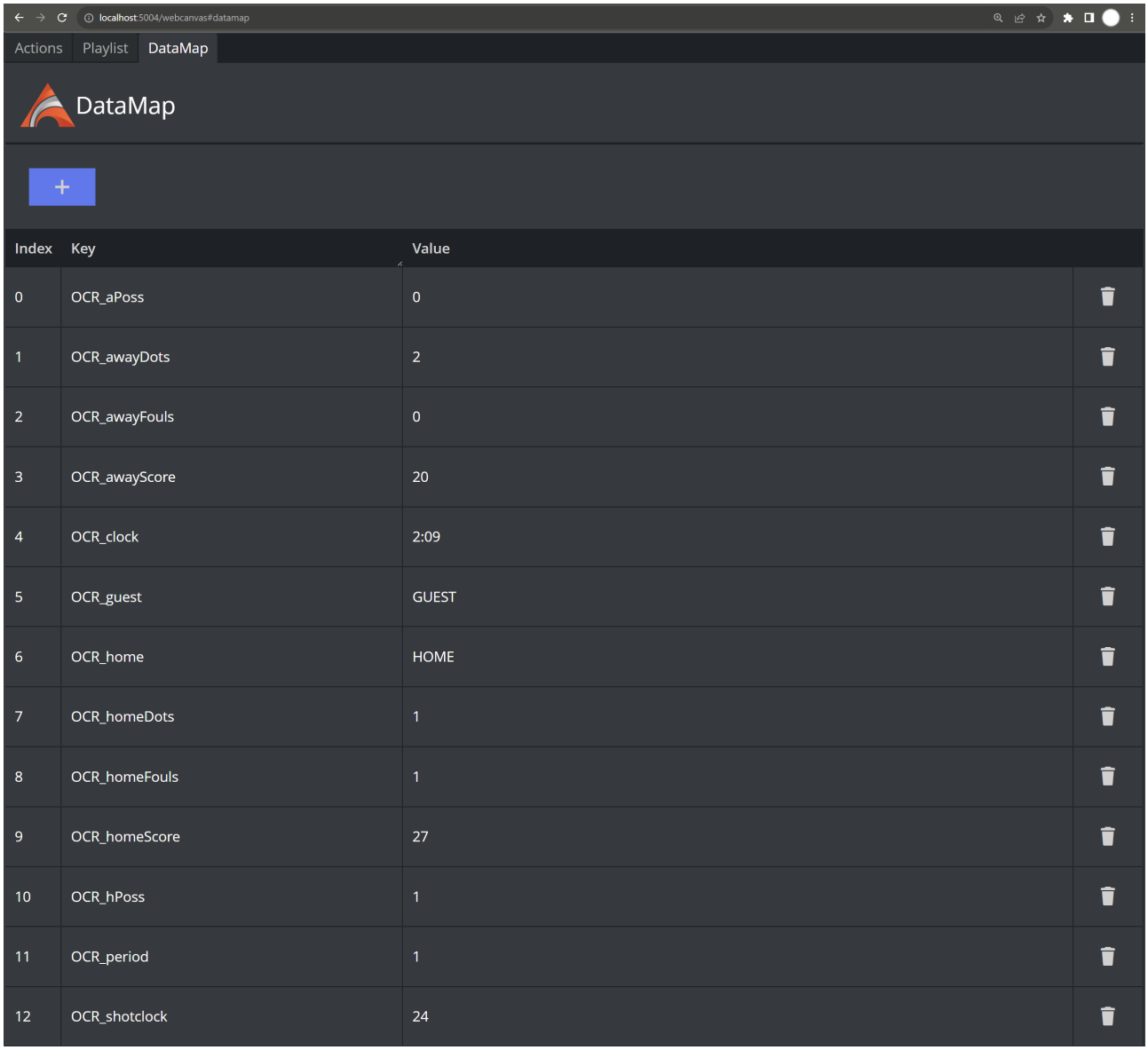
Example

This is how a fully configured widget appears.

Copyright © 2025 Vizrt



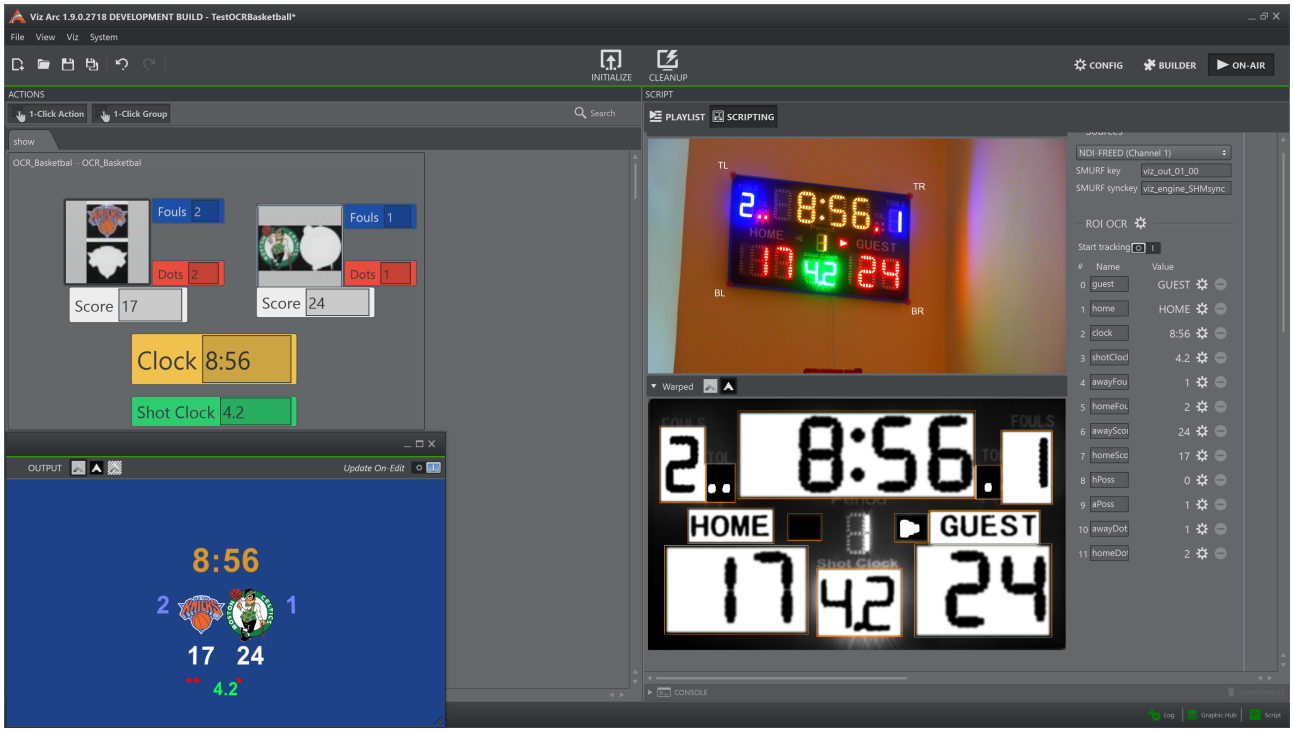
This is the relative DataMap:



Templates and DataMap

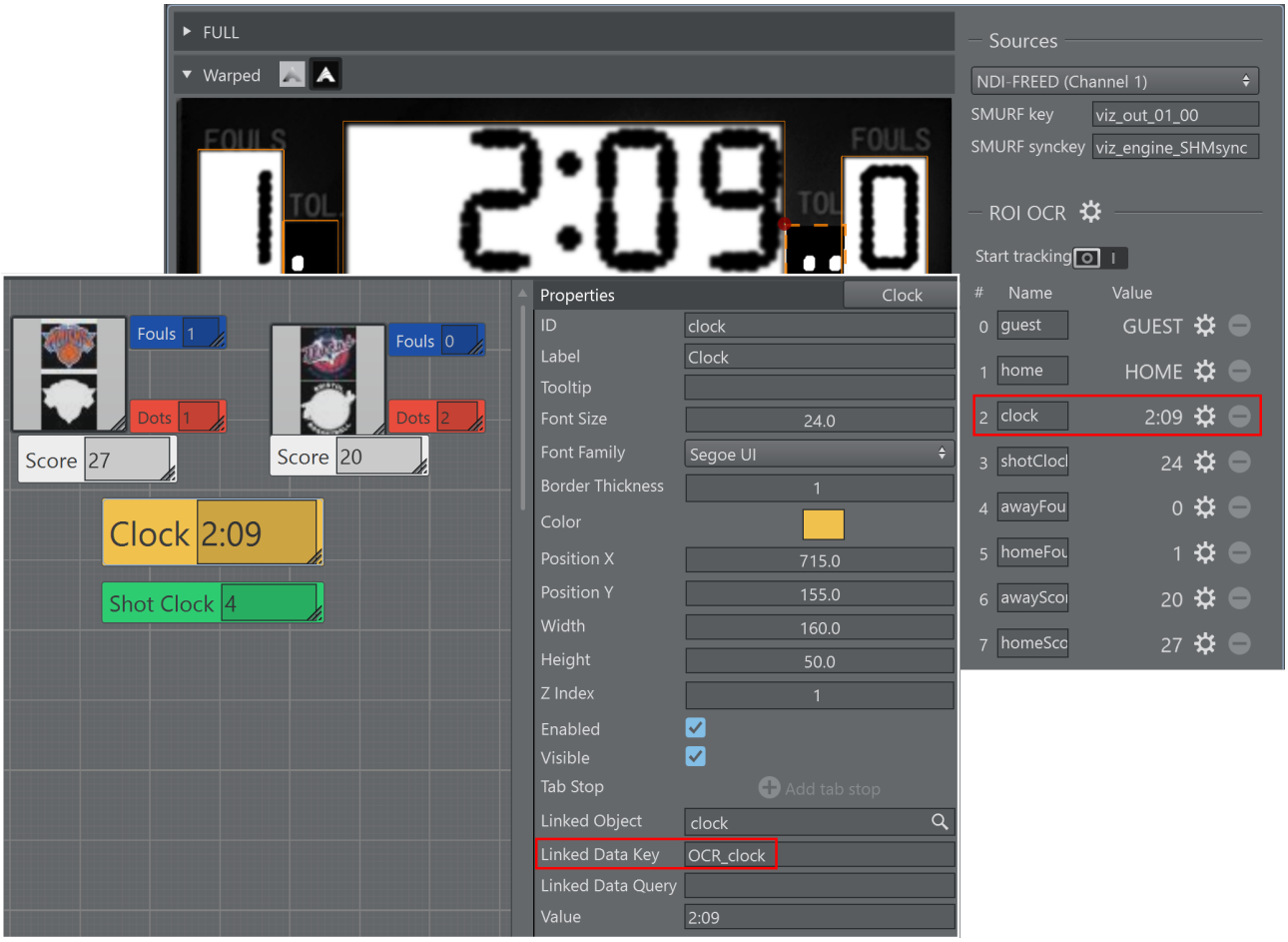
Thanks to the DataMap it is possible to link the results of the OCR to any template object as show below:

Copyright © 2025 Vizrt



This an example on how to link a template object to an ROI result:

Copyright © 2025 Vizrt

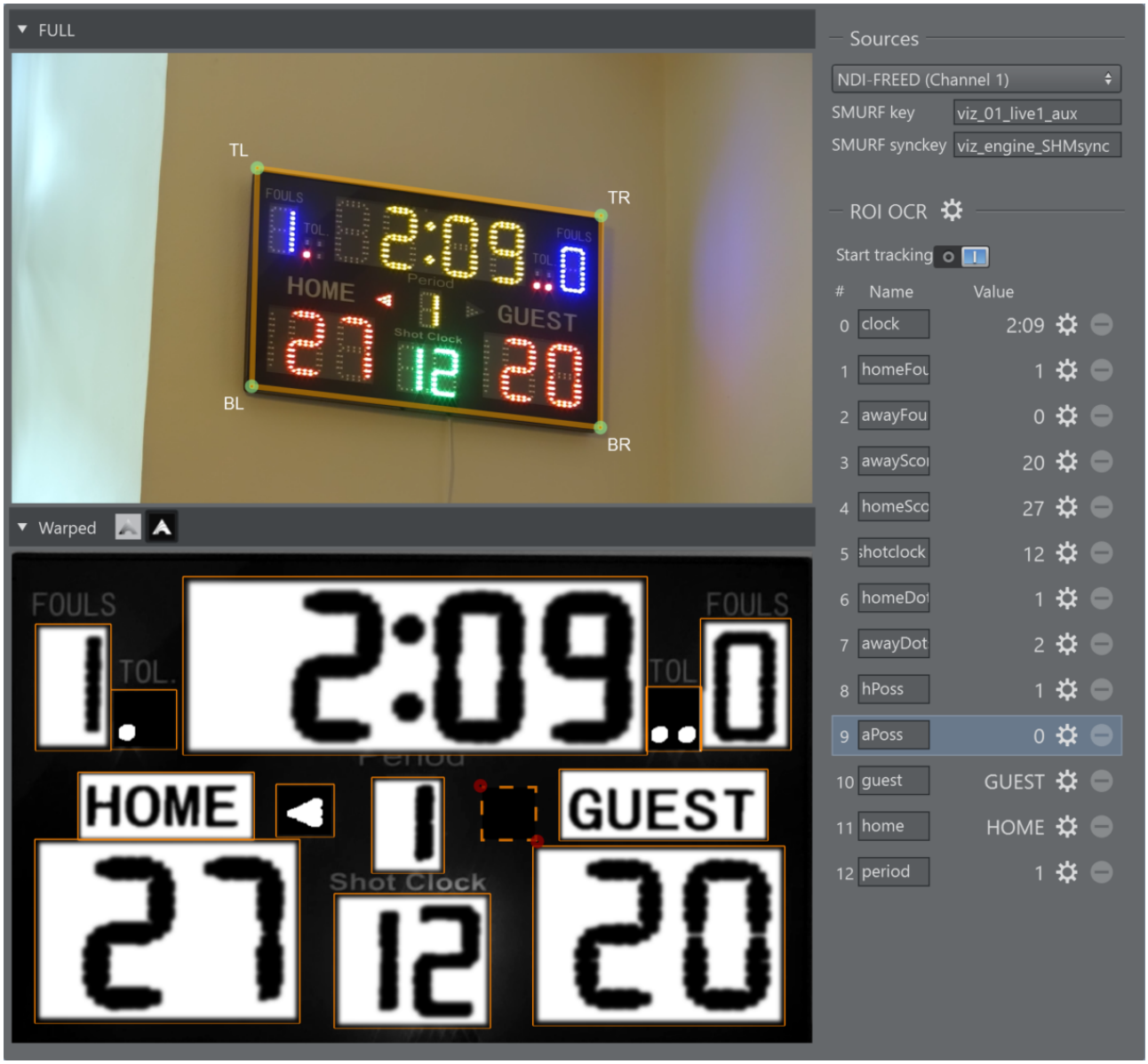


Tracking

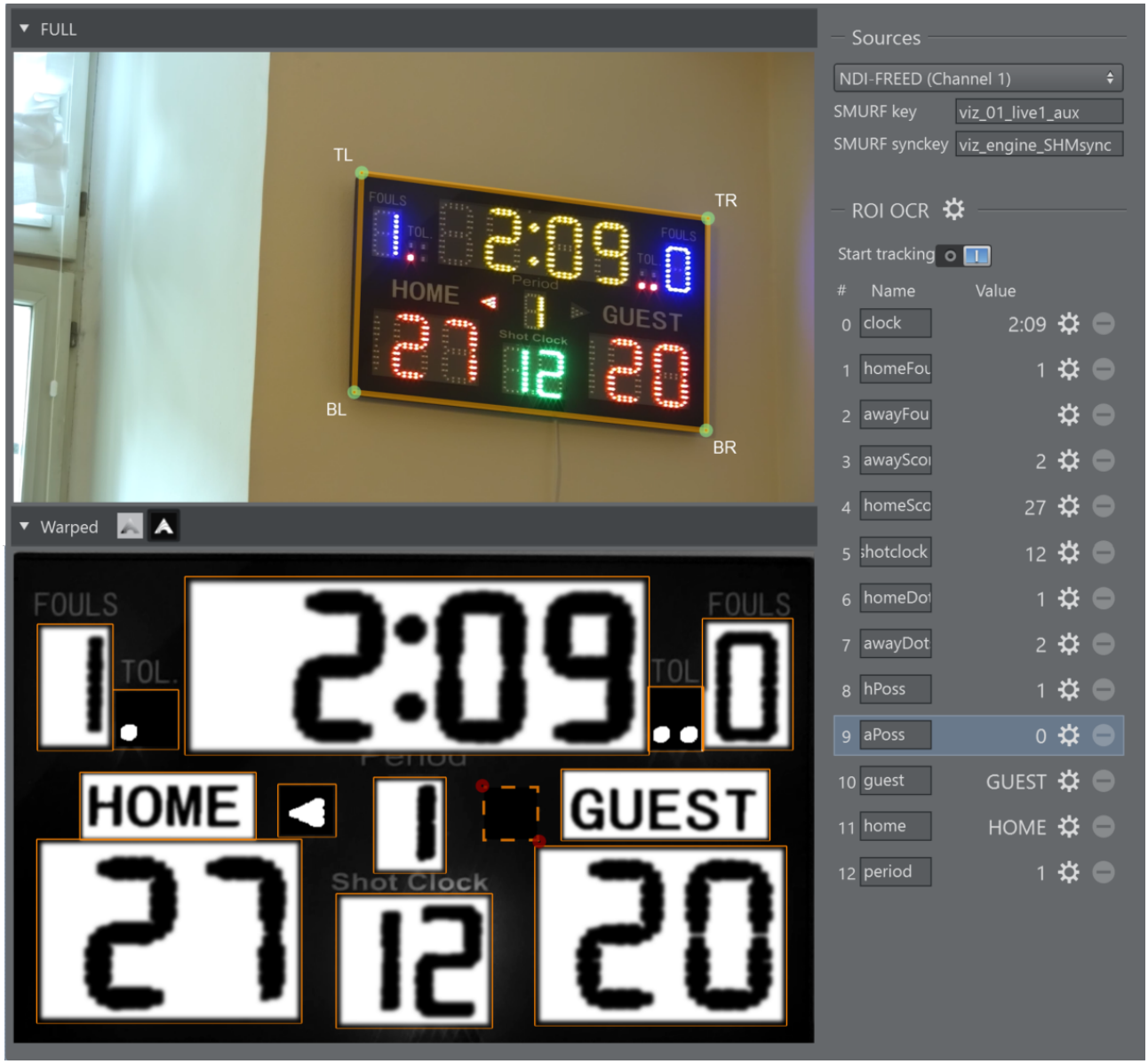
When the tracking is enabled, the algorithm tries to follow the target. Even if the camera moves, it continues to operate without the intervention of the user.

The quad changes color to confirm the tracking has started. While tracking is not possible to move/grab the control corners.

Copyright © 2025 Vizrt



Copyright © 2025 Vizrt


FULL
TL
TR
BL
BR
Warped
Sources
NDI-FREED (Channel 1)
SMURF key
viz_01_live1_aux
SMURF synckey
viz_engine_SHMsync
ROI OCR
Start tracking
Name Value
0 clock 2:09
1 homeFou 1
2 awayFou
3 awaySco 2
4 homeSco 27
5 shotclock 12
6 homeDot 1
7 awayDot 2
8 hPoss 1
9 aPoss 0
10 guest GUEST
11 home HOME
12 period 1
FOULS
TOL.
2:09
HOME
GUEST
Shot Clock
27
12
20

5.14 On Air Mode

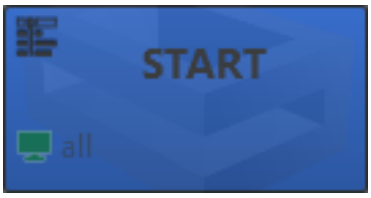
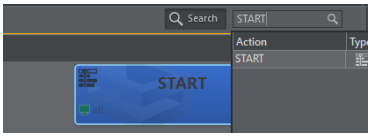
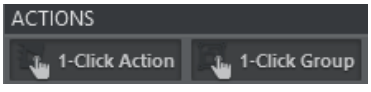
On Air mode lets you operate all the prepared actions in Builder Mode (ACTION, SET, SCRIPT and PLAYLISTS). Go On Air by clicking on the selected action (applies all single actions; the execute button is available on actions groups).

> **Note:** It's not possible to modify any aspect of the actions, all modifications have to be done in **Builder Mode**.

The running Script Form and Playlist section is displayed on the right side of the panel.

5.14.1 To Go On Air

This section shows how to operate in On Air mode:

	Once you are ready to go on air, **CLEAN UP** all Viz Engines connected to the selected profile.
	After cleanup, click the **INITIALIZE** button to load the scenes you have imported into the project.
	Once the scene is ready, you can easily use all the actions you have prepared in Builder mode.
	You can mark an action as a favorite during operation in order have it readily available.
	Find an action quickly be entering it in the search bar.
	Use the **1-Click Action** button to select whether you want **actions** to have direct **On Air buttons** or an **execute button** for each action, by adding an **execute** button in each action box. Use the **1-Click Group** button to select whether you want **group actions** to have direct **On Air buttons** or an **execute button** for each action, by adding an **execute** button in each action box.

Copyright © 2025 Vizrt

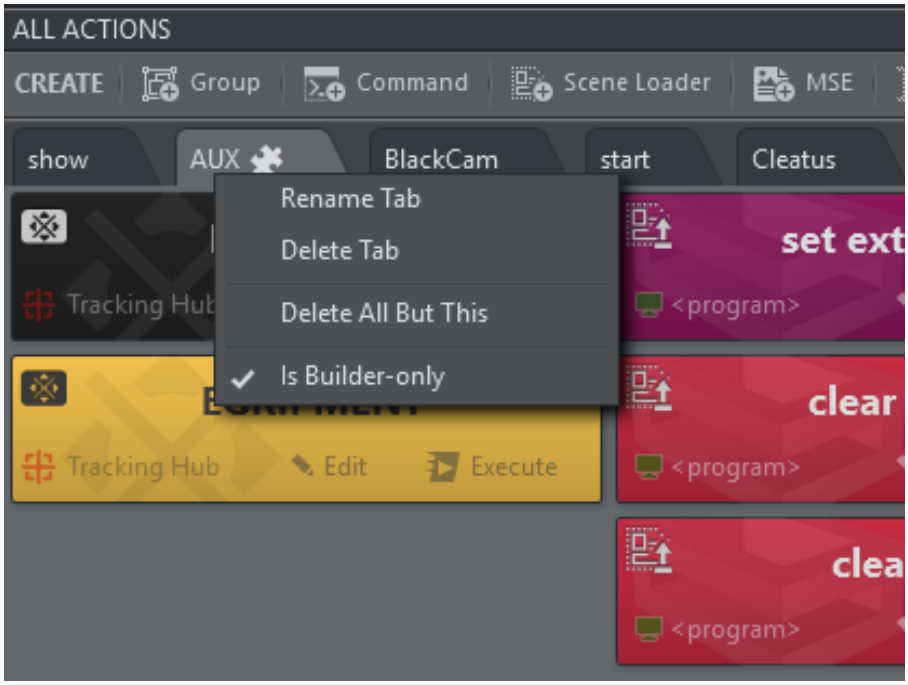
5.14.2 On Air Mode Desktop Shortcut and Command Line Switches



After installation, a desktop shortcut is created that's used to launch Viz Arc directly in On Air Mode, without the possibility of entering Builder Mode. You can use additional command line options for startup.

Note: Users can switch and *save as* projects in On Air Mode.

5.14.3 Builder-only Tabs

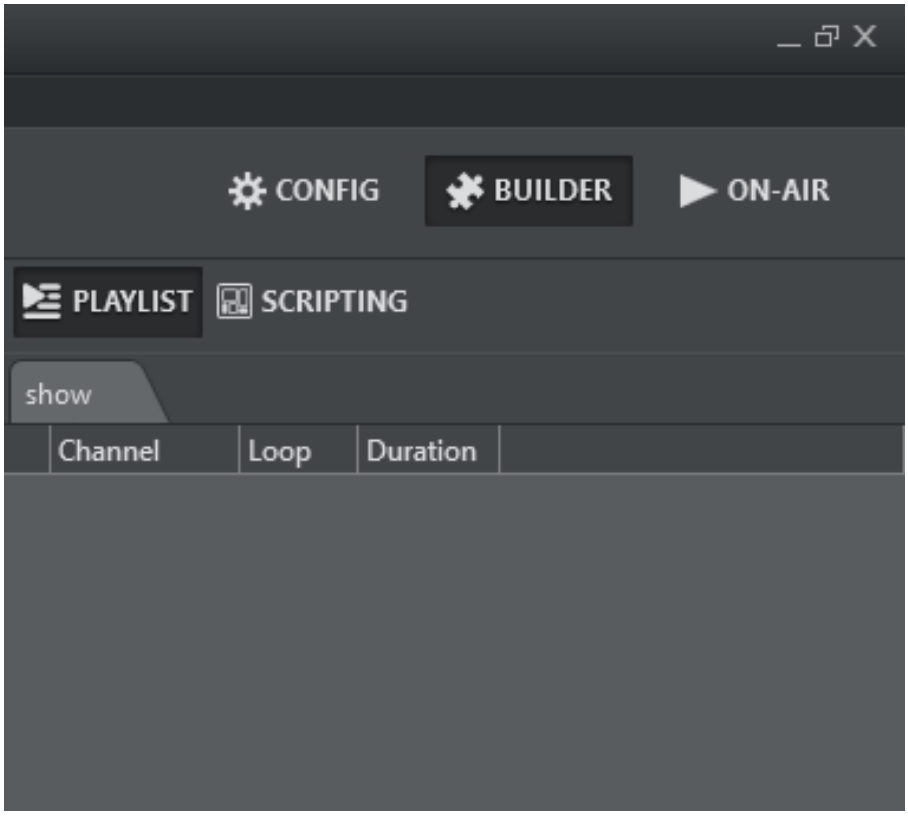


In **Builder Mode** you can right click on an action tab and select **Is Builder-only**. A small icon shows that the tab is **Builder-only**, the entire tab does not appear in **On Air Mode**. This prevents users from inadvertently executing actions that should not used during On Air operations.

Copyright © 2025 Vizrt

5.15 Playlist

You can add individual actions in a rundown and play the rundown either manually or automatically by defining timing on single playlist elements.

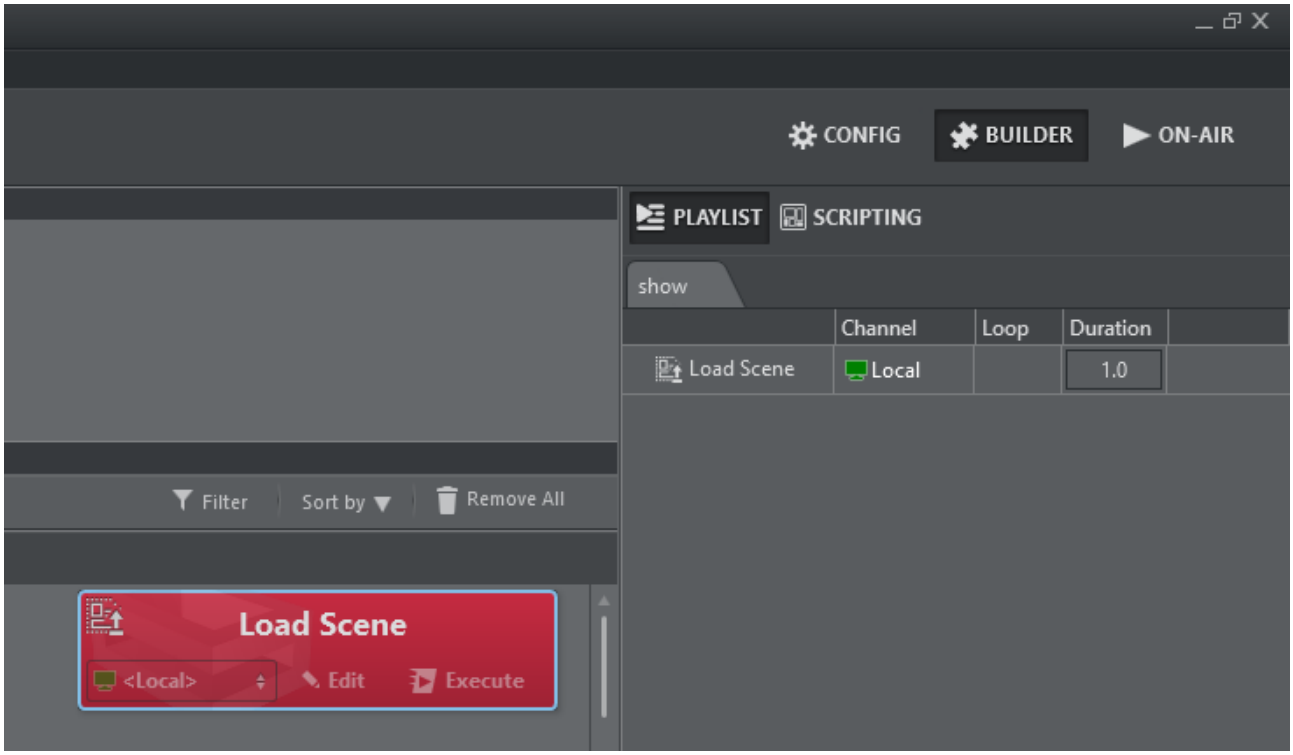


The **PLAYLIST** is at the top right of the UI in both Builder and On-Air mode.

5.15.1 Adding Actions to the Playlist

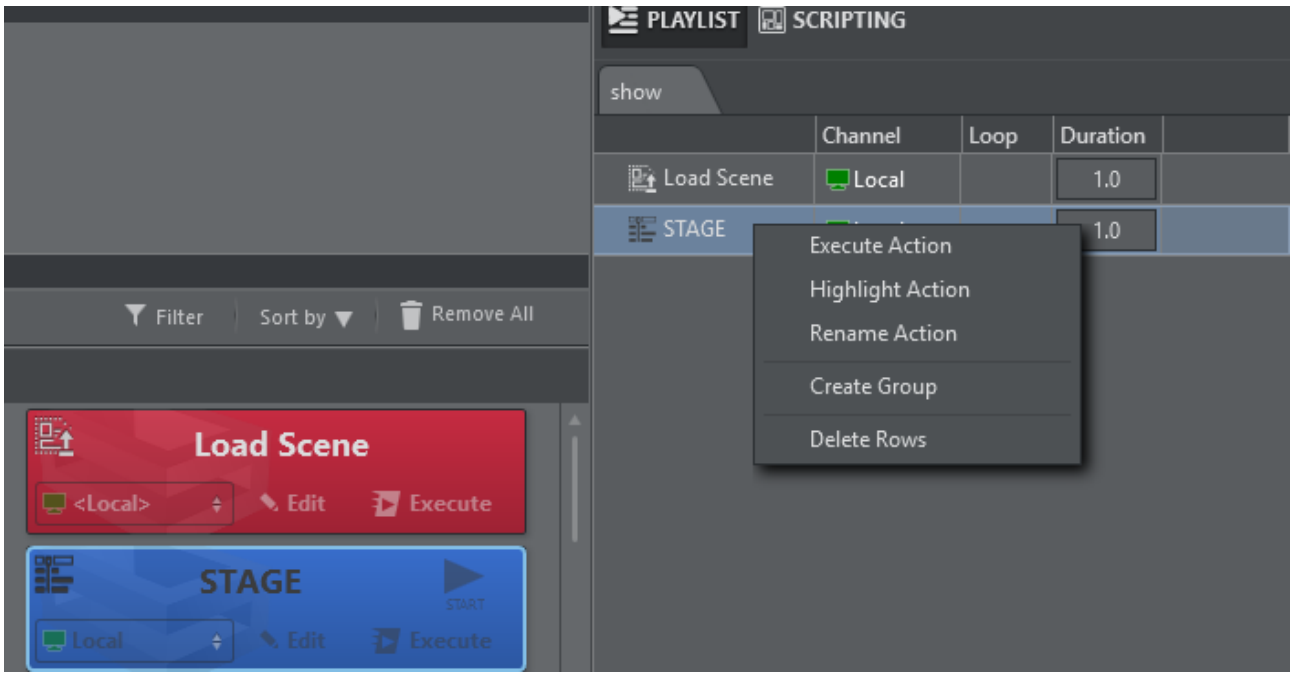
To add actions to the playlist, drag and drop an action from the action canvas into the playlist while holding in the **SHIFT** key.

Copyright © 2025 Vizrt



5.15.2 Working with the Playlist

Right-click on the playlist to make this context menu appear:

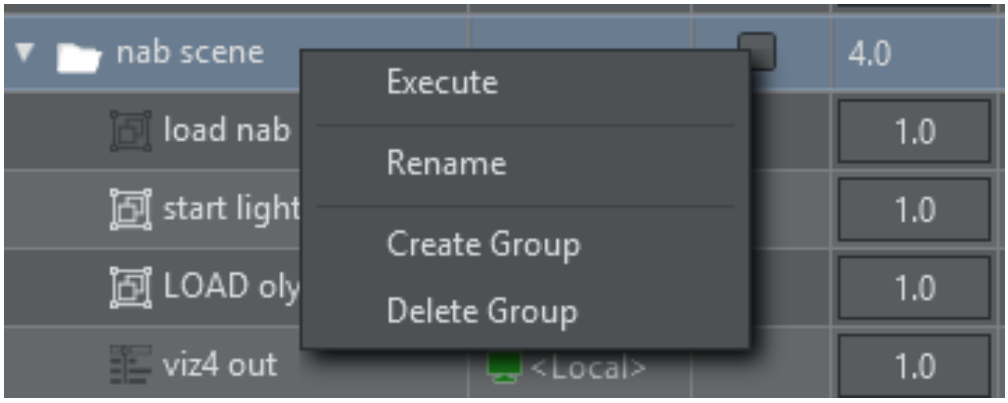


- **Execute Action:** Executes the currently selected action.
- **Highlight Action:** Highlights and brings into view the action on the action canvas.
- **Rename Action:** Renames the selected action.

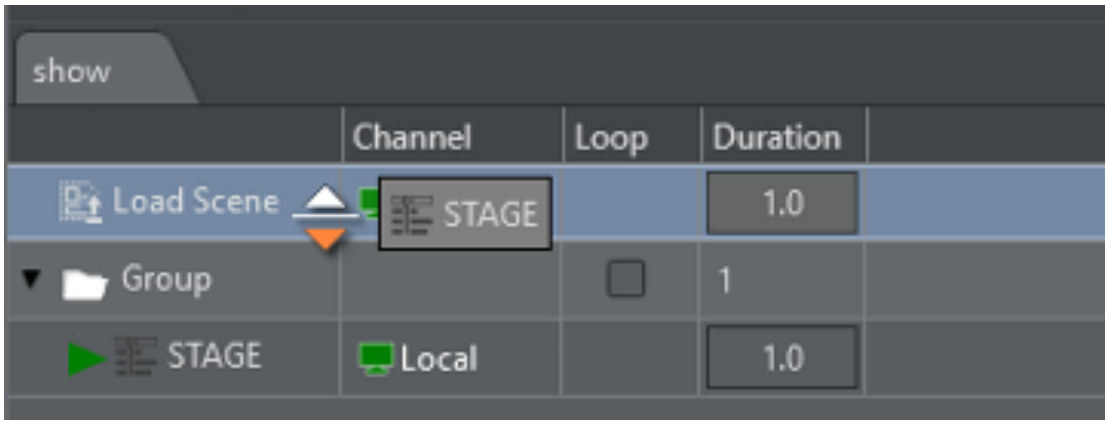
Copyright © 2025 Vizrt

- **Create Group:** Creates a logical group above the selected row.
- **Delete Rows:** Deletes the selected rows.

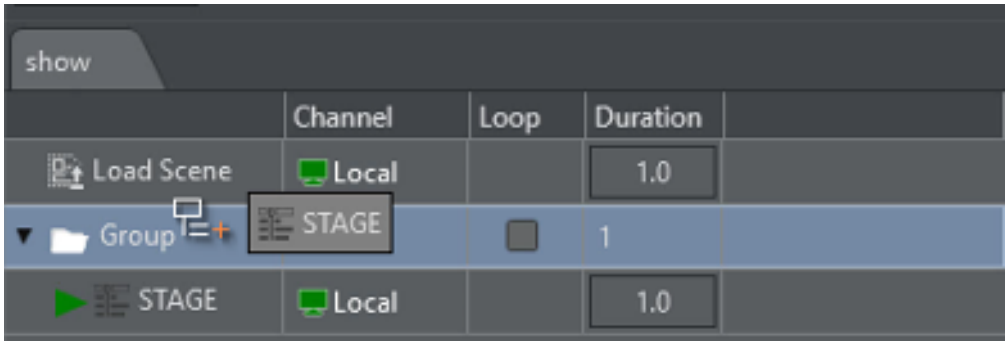
When opening the context menu on a **group** and selecting **Execute**, all children scenes are executed in order.



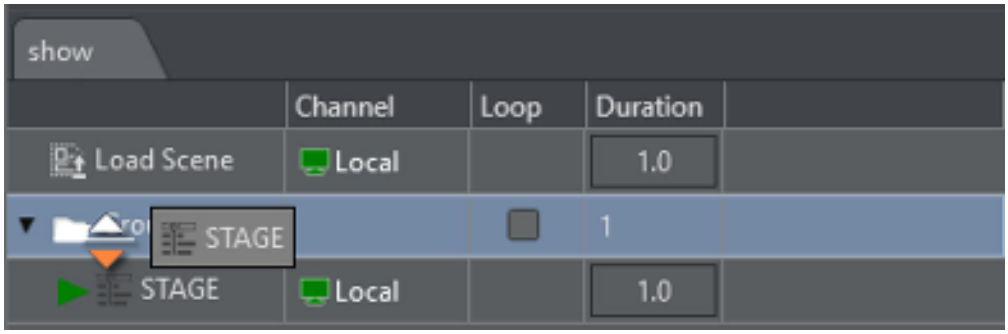
Use drag and drop to move actions around the playlist. The indicator shows if the action will be inserted before, after, or into a group.



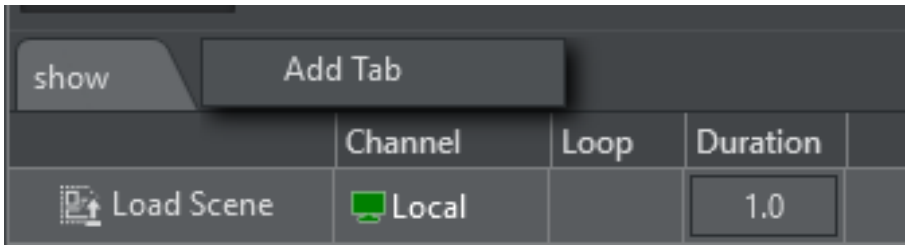
The **orange triangle** shows where the dragged item is dropped when released.

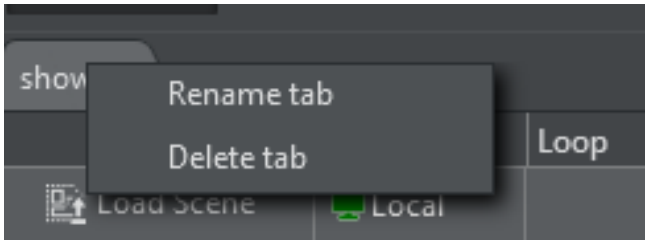


Copyright © 2025 Vizrt



- **Drop inside a group:** When dragging over a group item, the cursor icon shows whether the item is going to be dropped inside the group.
- **Drop under or above the group:** Moving the cursor further to the left allows you to drop under or above the group.
- **Add a new playlist:** Right click on an empty space in the playlist tab area.
- **Rename or delete a tab:** Right click on an existing tab.

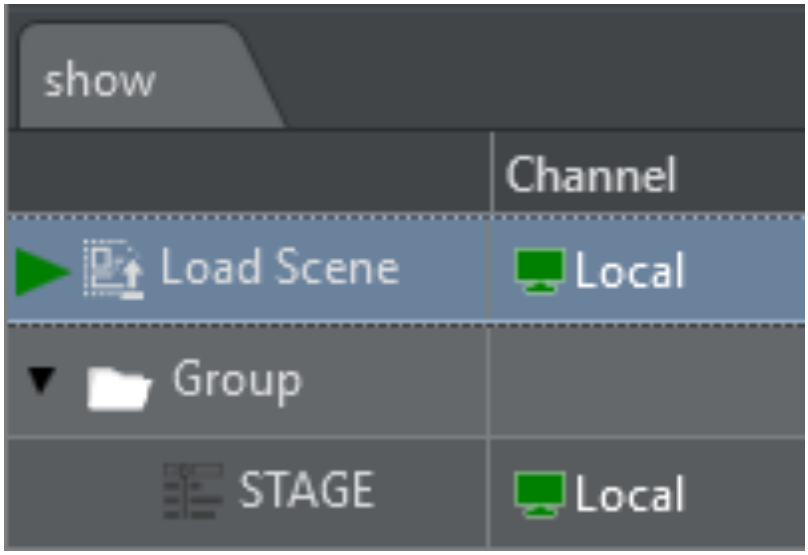




5.15.3 Using the Playlist

A selected playlist element can be executed using the **Execute Action** shortcut (default **F5**) or by double clicking the element. A separate system shortcut (default **ENTER** key) may be used to execute the current element and advance to the next element in the playlist. A **green triangle** shows the element that was executed most recently.

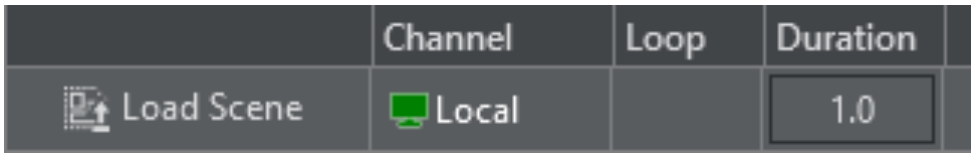
Copyright © 2025 Vizrt



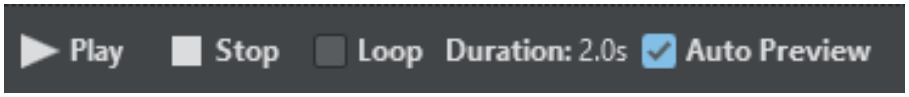
For template action items, the shortcuts for **continue, update, out** and **preview** work as well.

Duration and Loops

The very last column on a playlist element contains a duration (default 5 `.0`) representing the time in seconds that the playlist waits before executing the next element. The default duration can be changed under Playlist here.



Use the panel at the bottom to play the playlist:

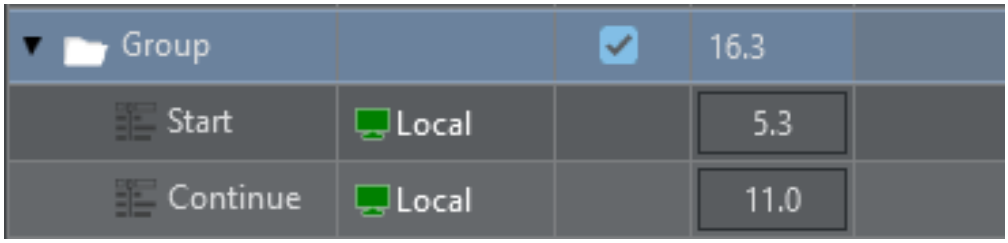


- **Play** starts the playlist from the top and **stop** interrupts playout.
- When **Loop** is checked, the playlist loops indefinitely.
- **Duration** indicates the duration of one single run.
- **Auto Preview:** If checked, any item selected on the playlist is executed on the preview channel.

Information: When the playlist has been scheduled for automatic playout, all modifications made on the playlist are ignored for the playout. The playlist must be stopped and played again in order to reflect any changes.

Groups can also be checked for looping. The elements within the group are then executed indefinitely until the stop button is pressed.

Copyright © 2025 Vizrt



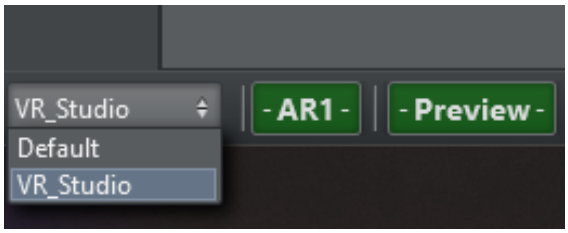
Copyright © 2025 Vizrt

5.16 Preview Studio

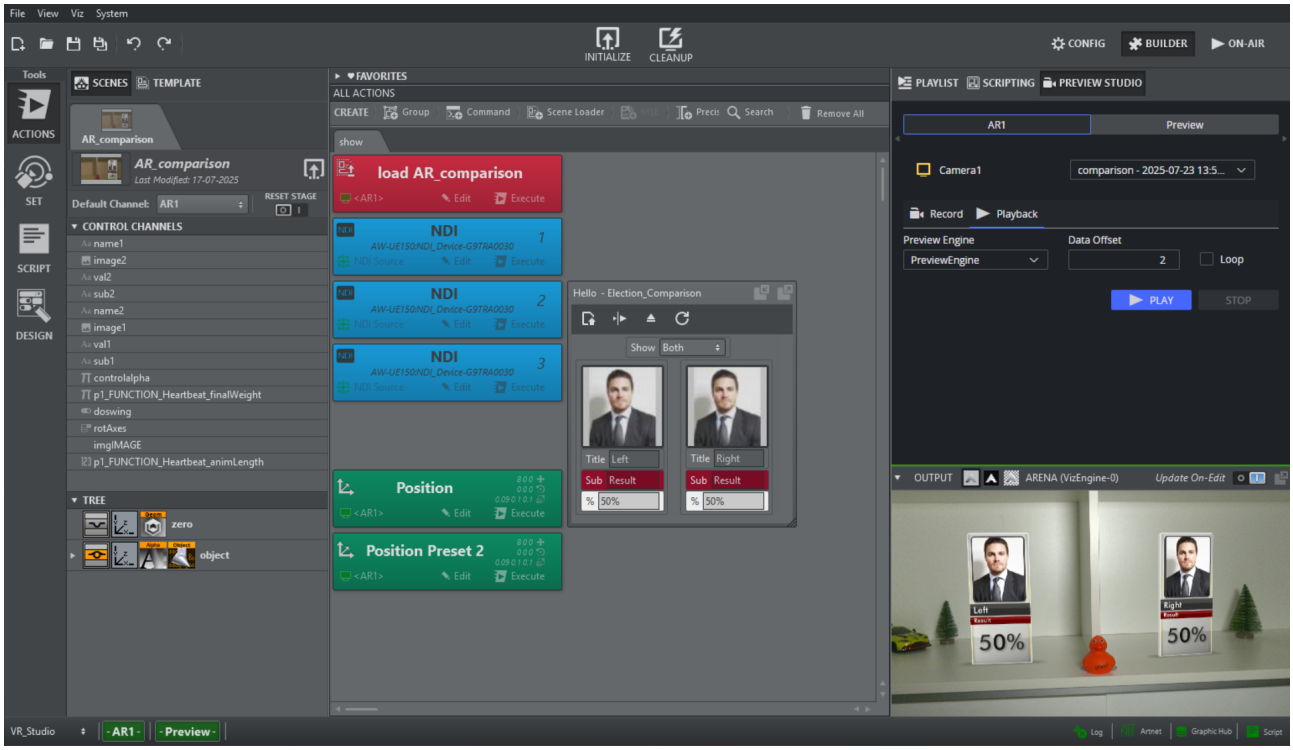
- Recording
- Playback
- Limitations

5.16.1 Recording

To start a recording, select the previously created profile from the bottom-left corner of Viz Arc.

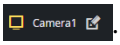


Select the **PREVIEW STUDIO** tab on the right sidebar.



In the Studio Preview UI, the status of the remote Engine Services and their status can be seen. For each Engine Service instance, the status on the remote machine and the available disk space, is shown. When the icon is yellow, the Engine Service is available and idle, when grey, it is not reachable, installed or running.

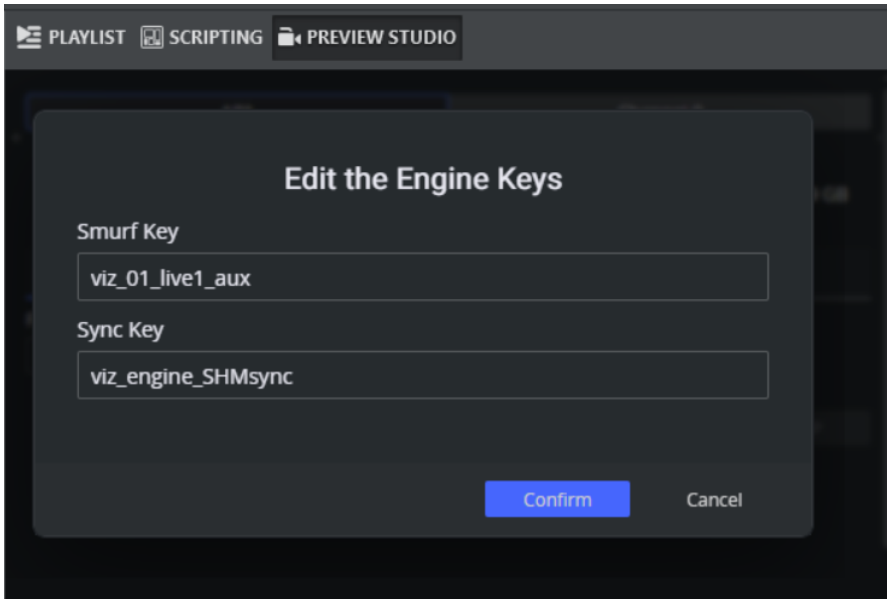
It is possible to switch between the different channels of the Profile, if more then one is available.

By hovering on an Engine Service instance the edit button appears 

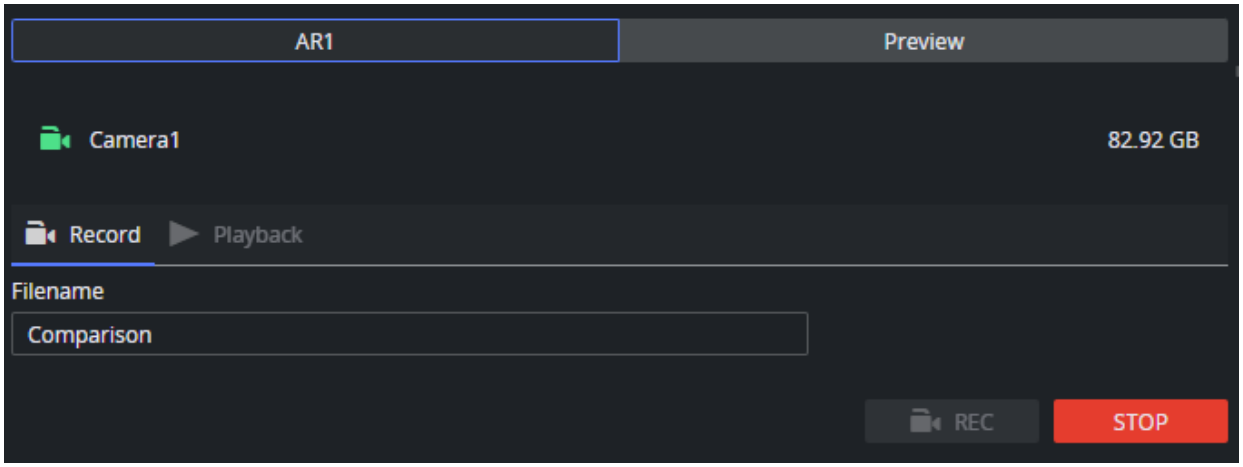
.

Copyright © 2025 Vizrt

By clicking on it, the **Smurf Key** and **Sync Key** can be modified. The keys indicate the input source from which the Engine service should record the video stream of the engine. The default values are set to grab the video from the live input 1 of the Viz Engine.



If the keys are correctly configured, enter a filename for the new recording and press **REC**. Make sure there is a valid scene loaded with a live input in texture mode, and configured with **SHM Aux Mode** as **Send**. Also make sure the preview engine is configured to receive and interpret the embedded tracking data of the incoming NDI stream.



Once the recording has started, the icon(s) of the Engine Service(s) turn green and the available disk space updates accordingly.

During the recording, it is not possible to switch to the **Playback** tab.

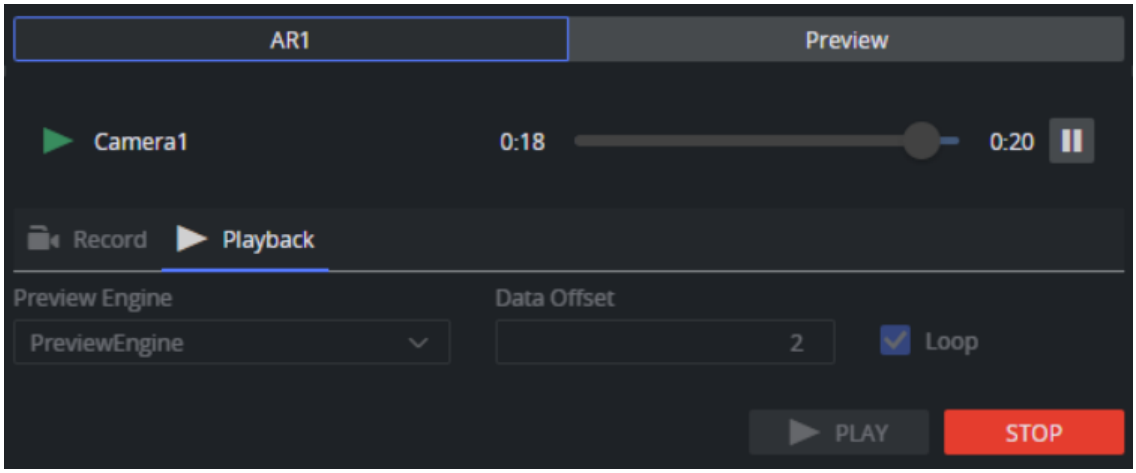
5.16.2 Playback

During the Playback, each remote instance of the Engine Service creates an NDI stream that carries both the video and the tracking data. This stream can be set as a Video Input into the Viz Engine that is going to act as Preview Server.

Copyright © 2025 Vizrt

To start a Playback, click on the Playback tab and select the desired recording from the dropdown list. Each instance of the Engine Service can stream a different recording. It is also possible to choose which engine is used as the Preview from the **Preview Engine** dropdown list.

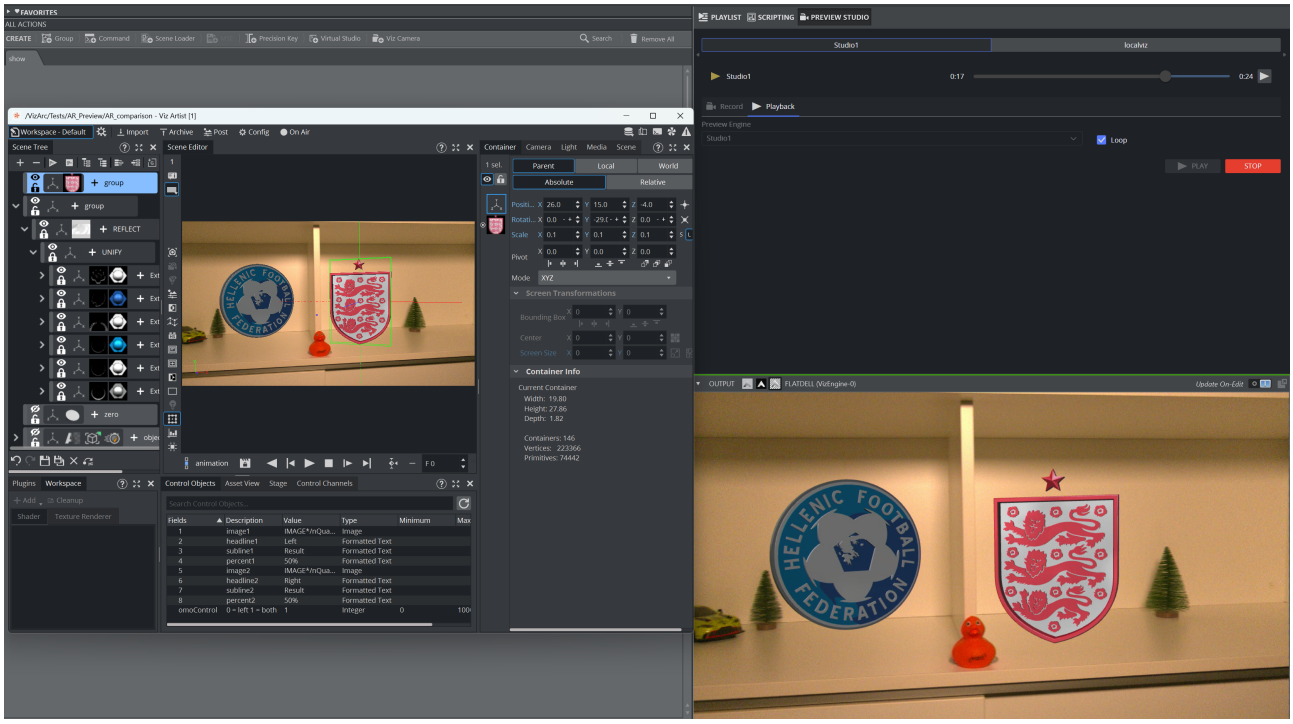
If **Loop** checkbox is ticked, once the playback is over, it starts back from the beginning.



By clicking on the Play button, the NDI streams are created and set as the input source of the specified Preview Engine. A slider shows the current position of the clip and can be used to seek to a certain position. The play/pause button can be used to stop and resume the player. The **Data Offset** parameter represents a offset (in fields), between camera tracking data and video during playback towards the preview engine.

An artist can now work on the preview engine in Artist mode using the same scene that is used during recording, or any other scene where he can fine tune and test graphics without having to rely on a fully started and set-up studio environment.

Change colors, adjust geometry that causes z-fighting, or test a completely different graphics:



Copyright © 2025 Vizrt

5.16.3 Limitations

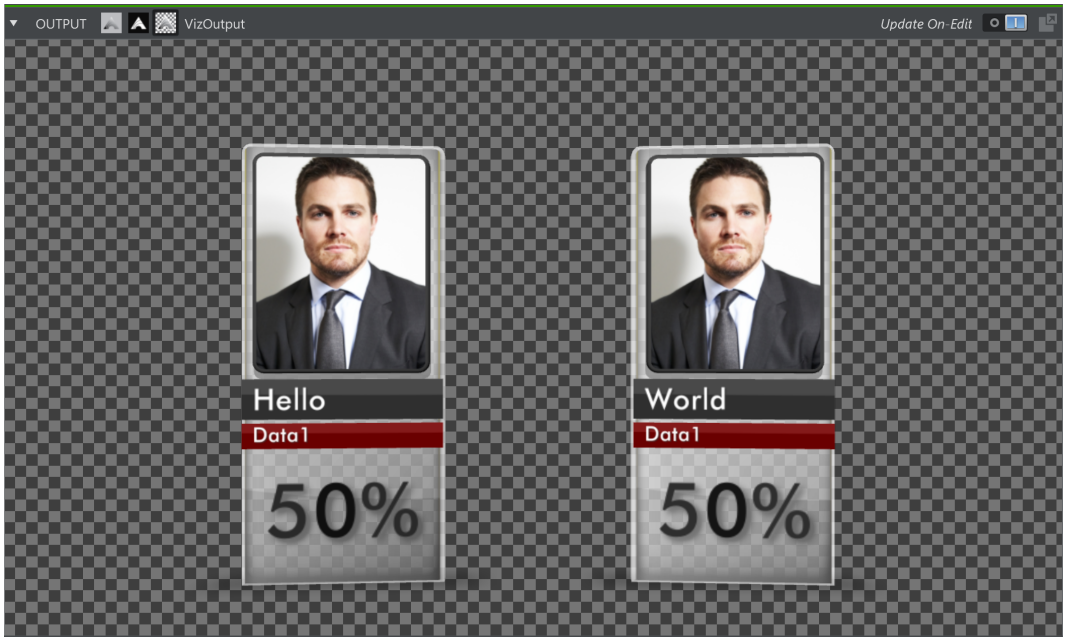
Once the tracking is recorded, it cannot be modified (for example, it is not possible to add a pan offset or change lens distortion parameters of the recorded tracking).

Using the playback feature might impact the performance of the engines playing out the NDI stream. It is not recommended to use this feature while the Viz Engines are on-air during a live production.

Copyright © 2025 Vizrt

5.17 Preview

This pane displays the Viz Engine output or any other video source configured. By enabling **Update On-Edit**, it is possible to see project changes in real-time.

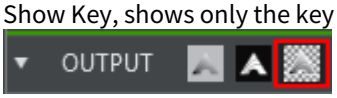


Use the three icons at the top left of the preview panel to select a visualization mode for the preview.



1. 

   Show Fill, shows only the fill channel.
2. 

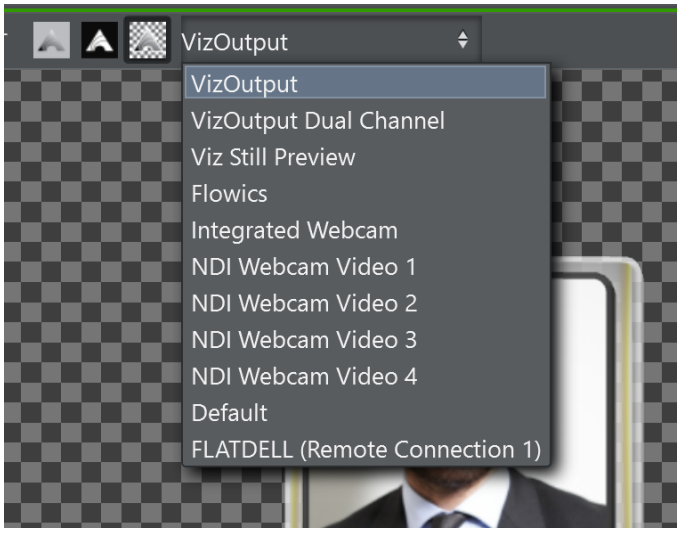
   Show Key, shows only the key channel.
3. 

   Show Keyed, shows only the keyed output.

Information: Note that currently only NDI sources and the Viz Output through shared memory (starting form Viz Engine 4.x) are capable of carrying fill and key information. This feature only works on NDI and Viz Engine sources. Select **Show Fill** for any other sources.

A drop down menu on the right side of the output mode buttons, can be used to quickly change the source of the output among the available outputs.

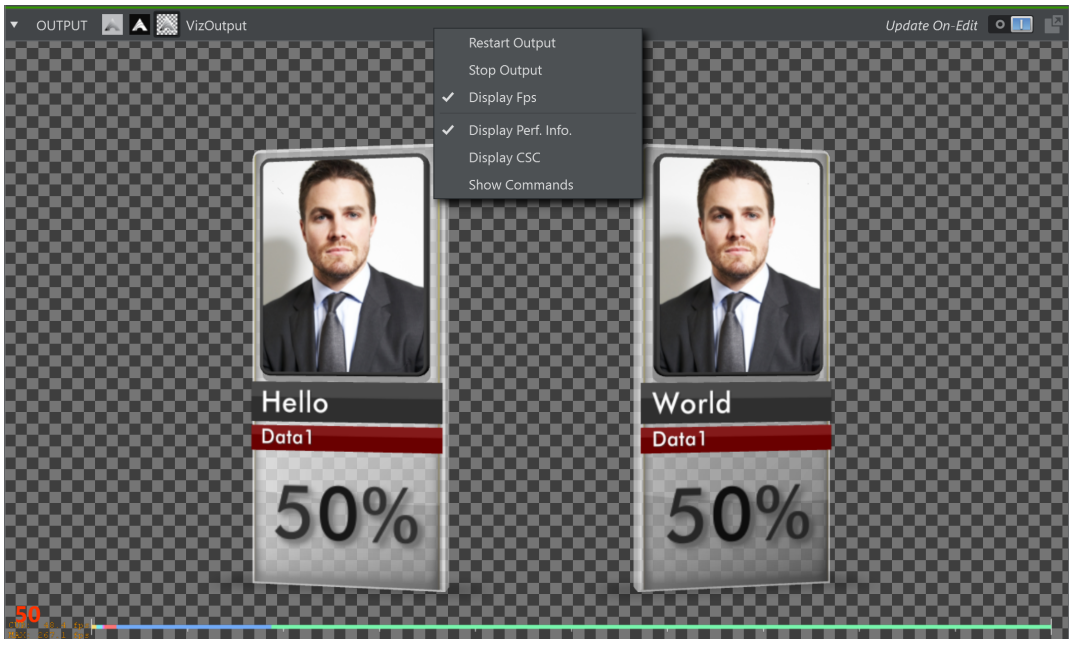
Copyright © 2025 Vizrt



Click the **Detach** button to pop out the preview window to a separate window. Close the window again to attach it back into the main window.



Use the top bar context menu to add additional useful overlays.



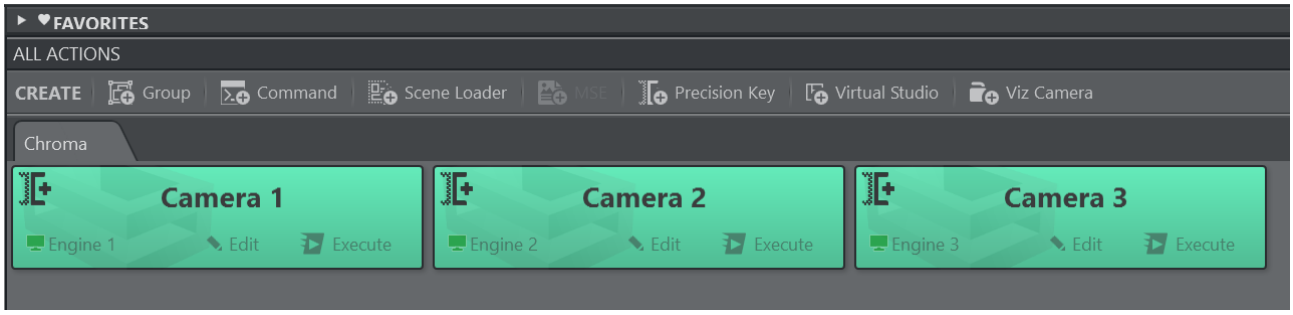
- **Restart Output:** Restarts the stream decoding process.
- **Stop Output:** Stops the output, might be used to save CPU/GPU resources.
- **Display Fps:** Displays the effective frames per second of the actual display in Viz Arc as a red number on the bottom left corner.
- **Display Perf. Info:** Shows the bottom performance bar when connected to a Viz Engine.
- **Display CSC:** Displays the center cross when connected to a Viz Engine.
- **Show Commands:** Shows commands when connected to a Viz Engine.

Copyright © 2025 Vizrt

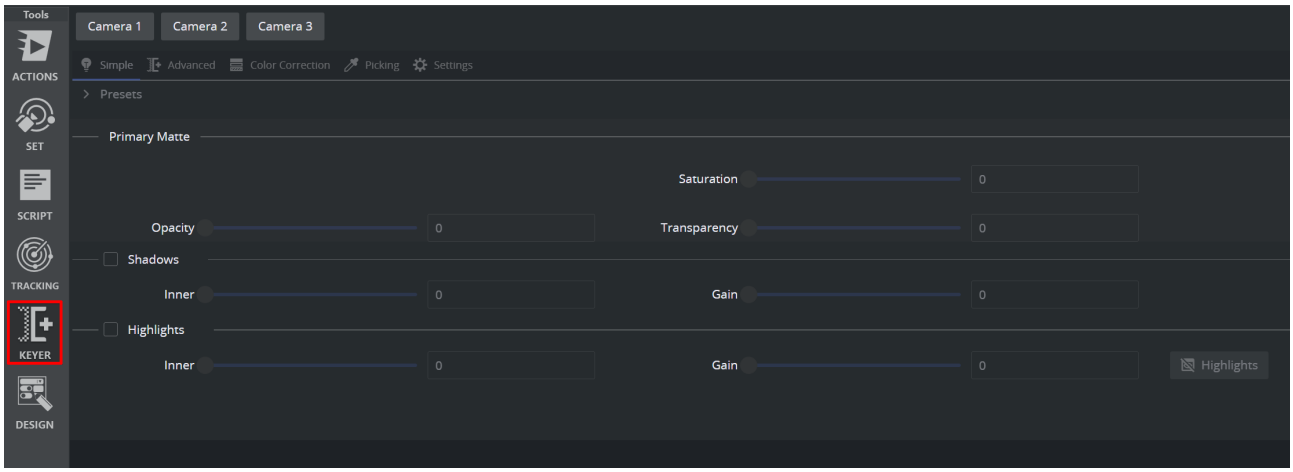
5.18 Precision Keyer

- Simple
- Advanced
- Color Correction
- Picking
- Settings
- Presets
- Execution
- Web View

The Precision Keyer can be controlled as usual through the standard UI. Additionally, there are other options to control when creating Precision Keyer actions.



Once at least one Precision Keyer Action has been created on the action canvas, the left hand tool bar displays a **KEYER** button which holds a dedicated UI for Keyer actions.

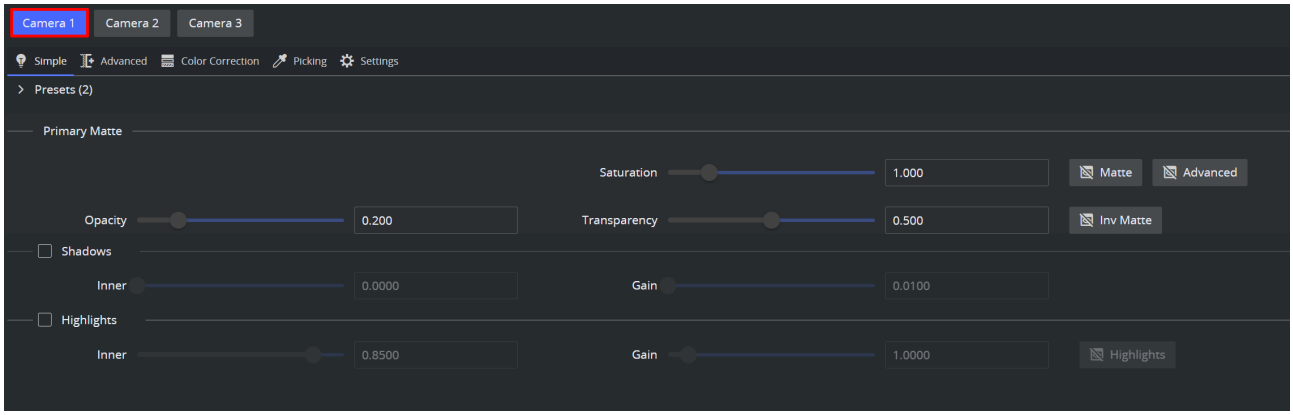


A list of all Precision Keyer actions is displayed on the top bar, click on on of them to work on it.

5.18.1 Simple

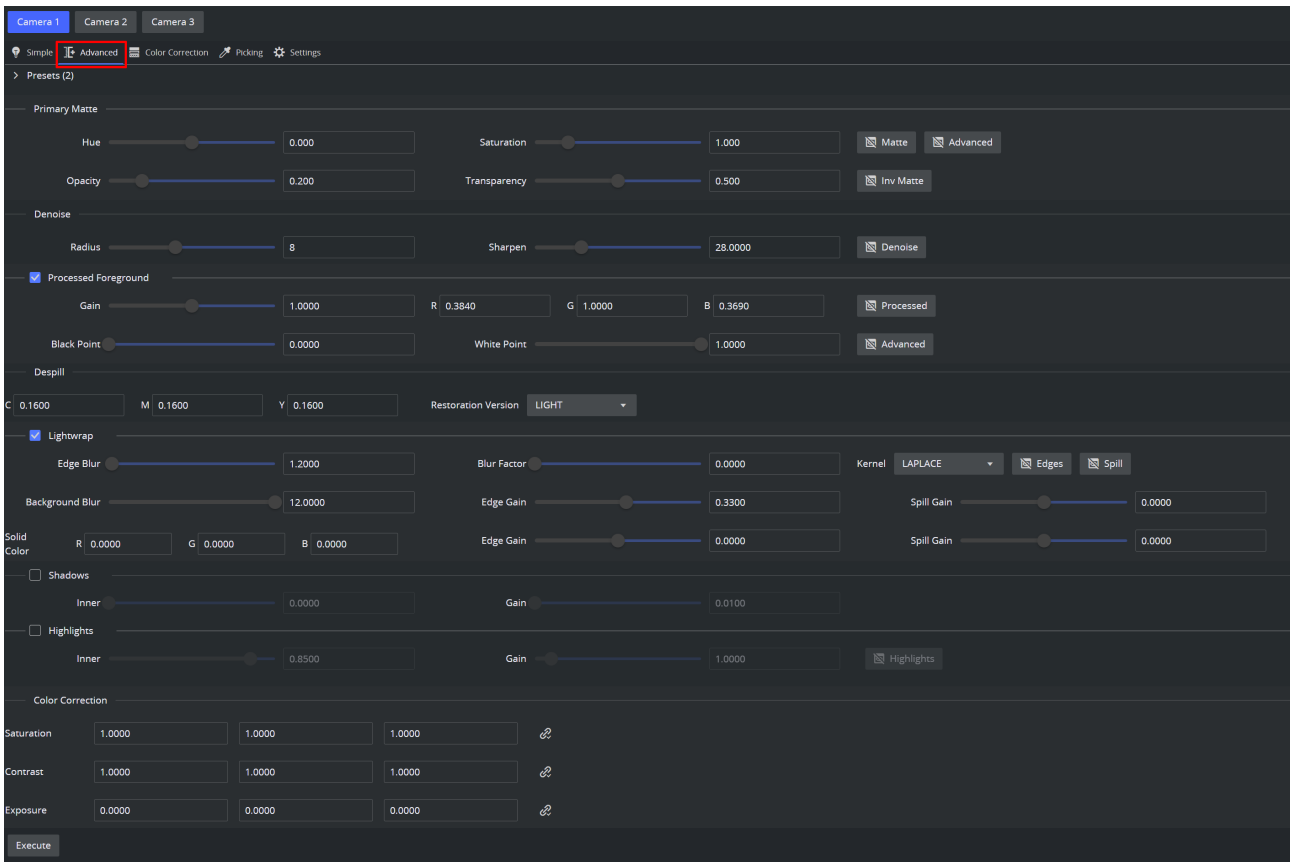
The **Simple** view displays only the most important settings that might be changed during On Air operations.

Copyright © 2025 Vizrt



5.18.2 Advanced

The **Advanced** view displays all values of the Precision Keyer.



5.18.3 Color Correction

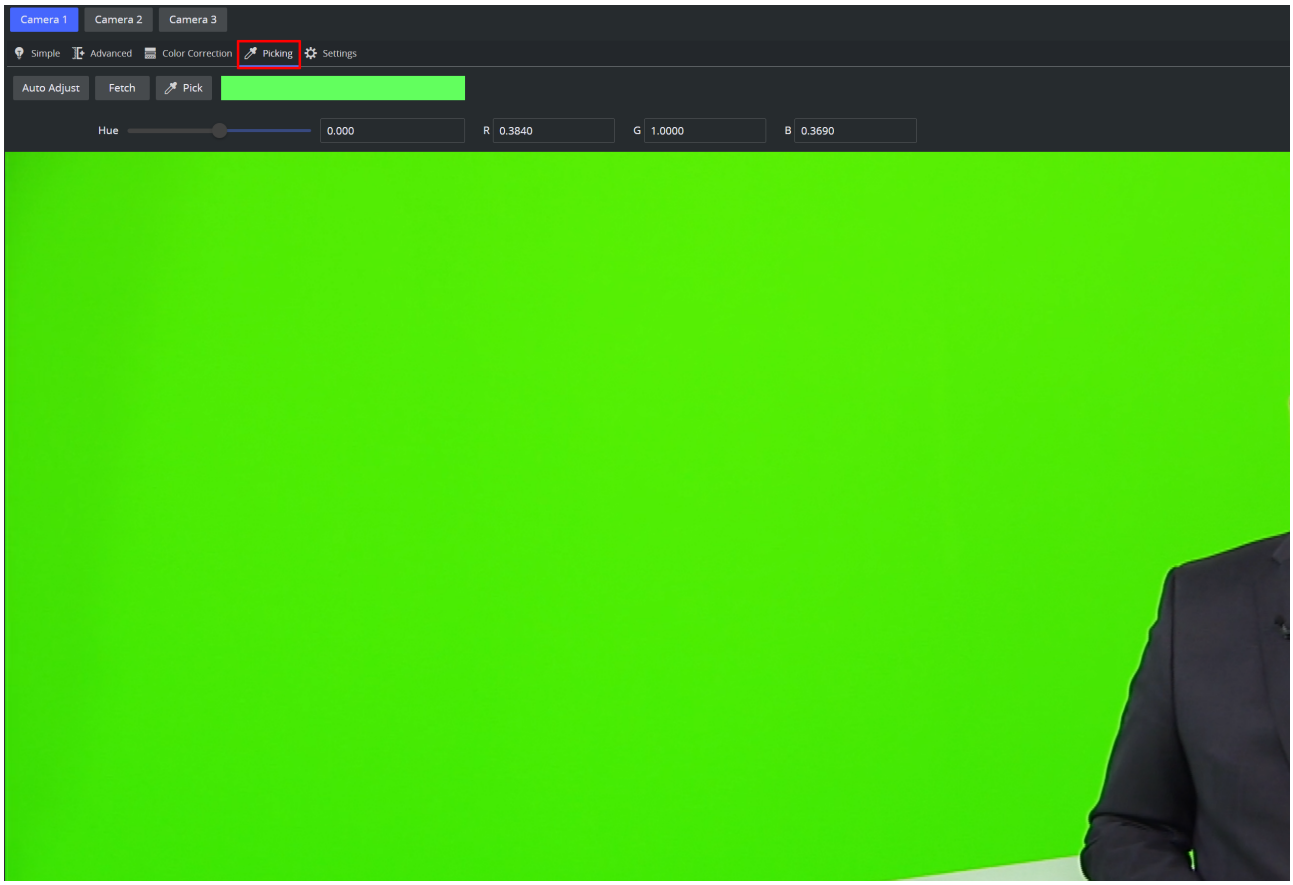
The **Color Correction** view displays the the color correction values with additional sliders.

Copyright © 2025 Vizrt



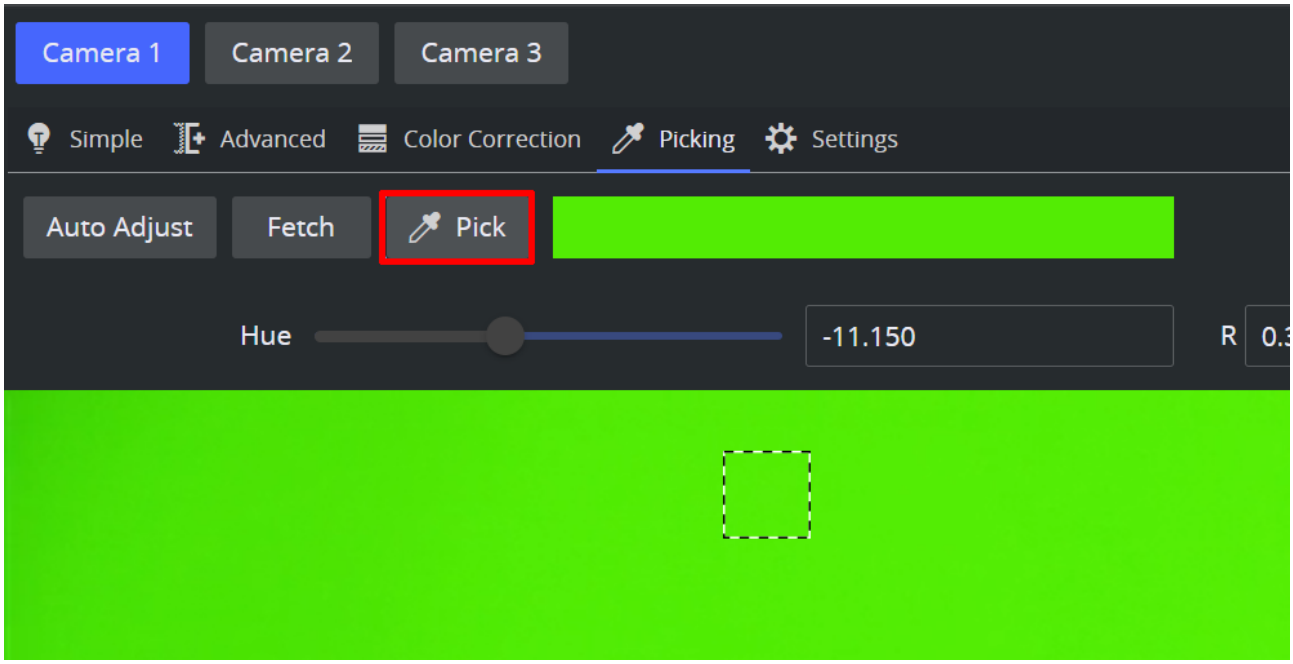
5.18.4 Picking

The **Picking** view shows the values relevant for the backdrop color and the hue value.

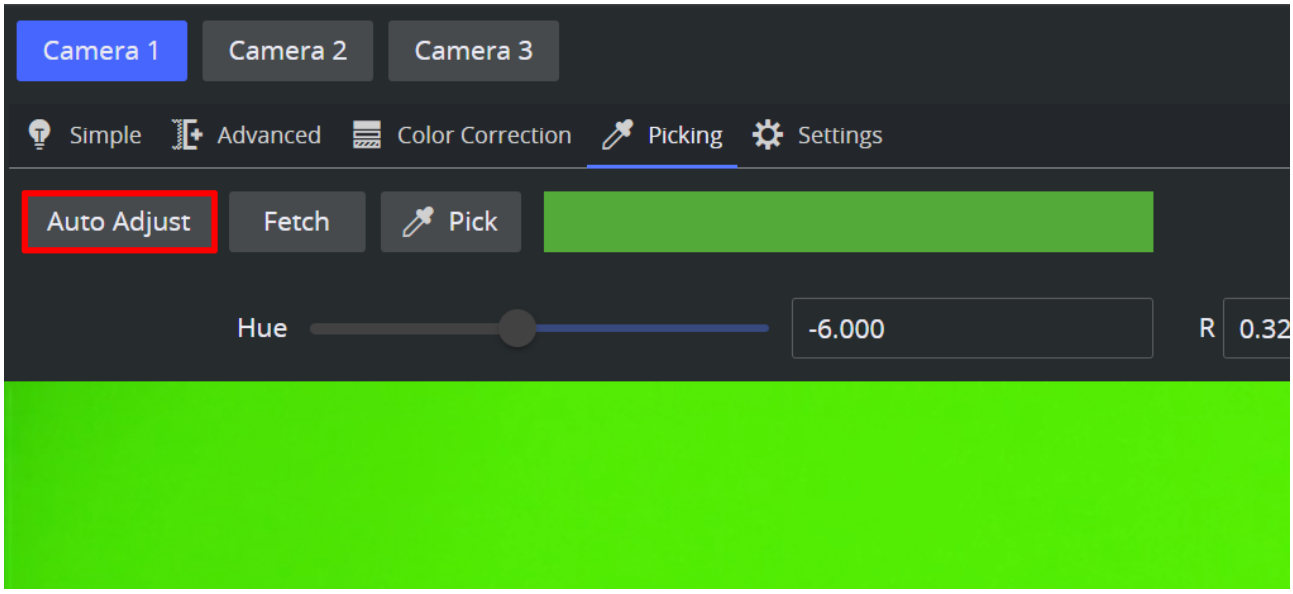


Use the **Pick** button to pick any color on the UI. Use the mouse wheel to increase or decrease the picking area.

Copyright © 2025 Vizrt

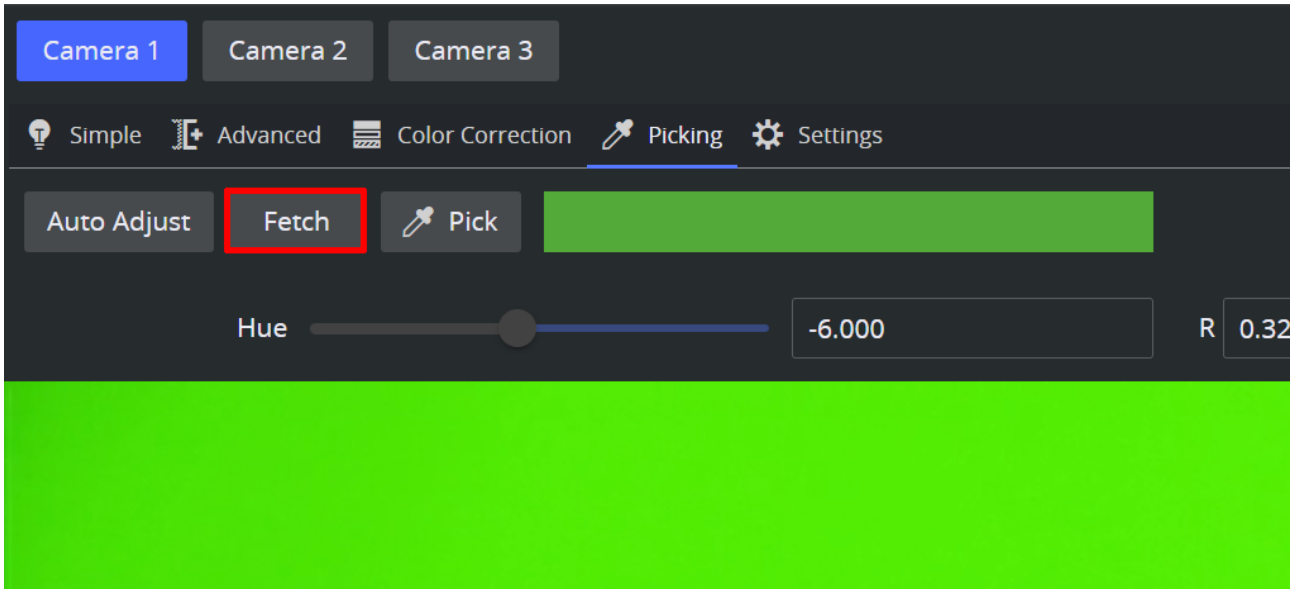


Use the **Auto Adjust** button to let the Viz Engine automatically determine an optimal backing color and hue.



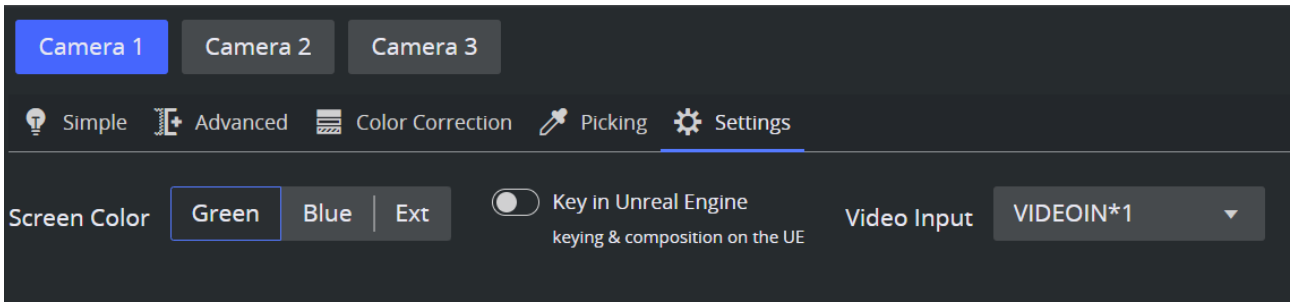
Press the **Fetch** button to get the current value from the connected Viz Engine.

Copyright © 2025 Vizrt



5.18.5 Settings

The **Settings** view shows the basic settings of the Precision Keyer.

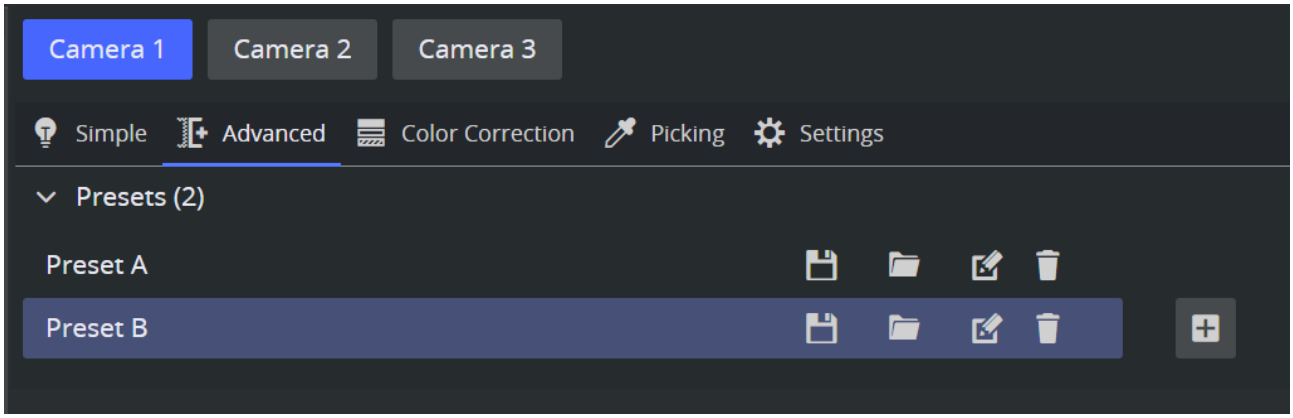


- **Screen Color:** Whether to key on a green or bleu screen or whether the keying should take place externally.
- **Key in Unreal Engine:** Whether the keying process should be calculated in the Unreal Engine or the Viz Engine.
- **Video Input:** The video source within the Viz Engine where the keying should take place.

5.18.6 Presets

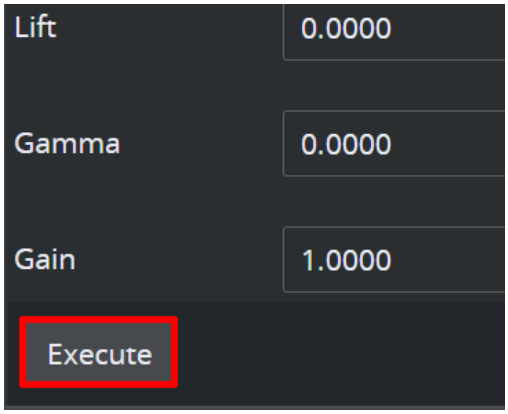
The presets can be recalled, stored, deleted and created like in the classic User Interface. The presets are available in the **Simple**, **Advanced** and **Color Correction** views.

Copyright © 2025 Vizrt



5.18.7 Execution

The entire chroma actions can be explicitly executed using the **Execute** button on the bottom left part of the view.

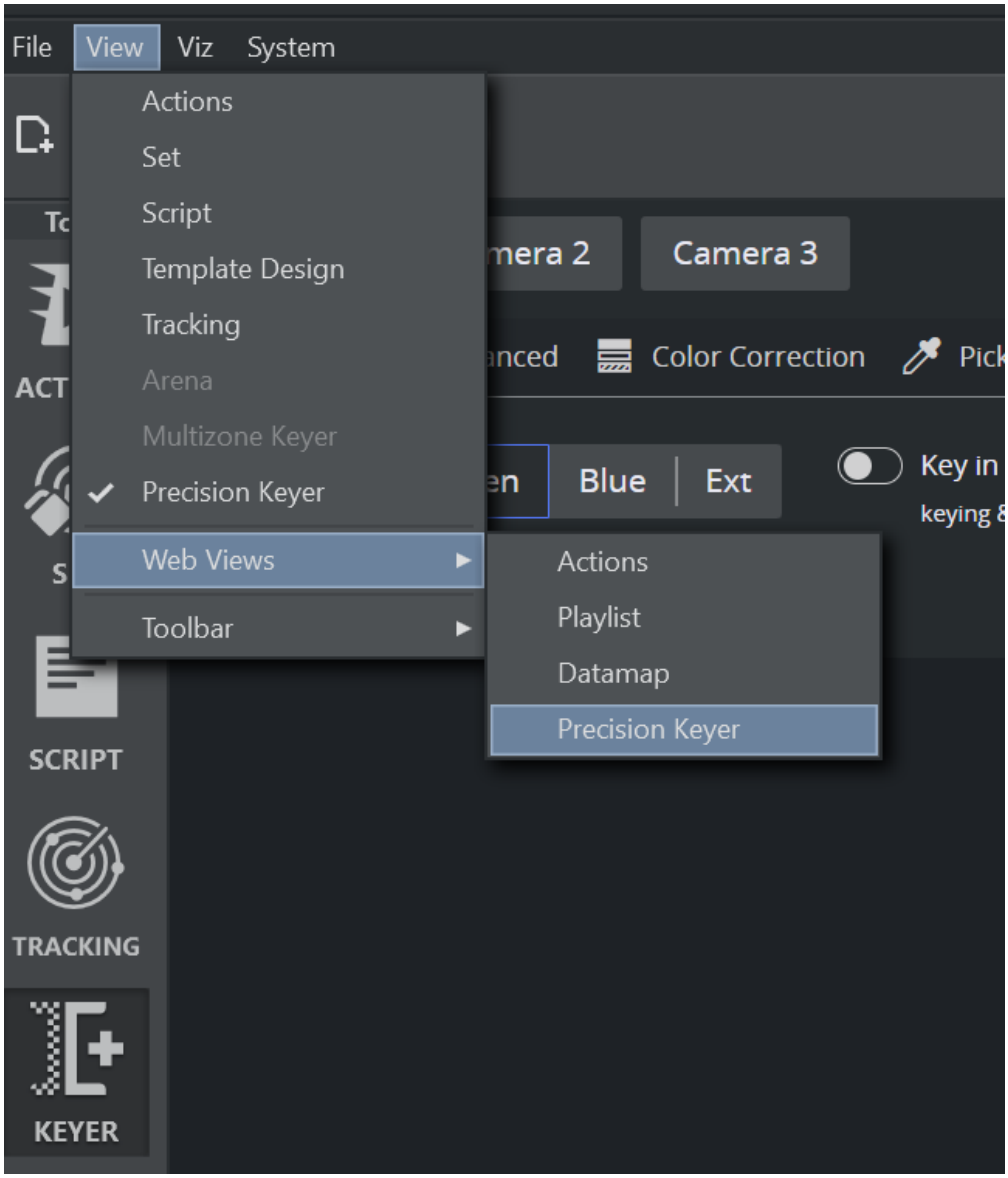


Note: Selecting the action does not execute the action on Viz Engine.

5.18.8 Web View

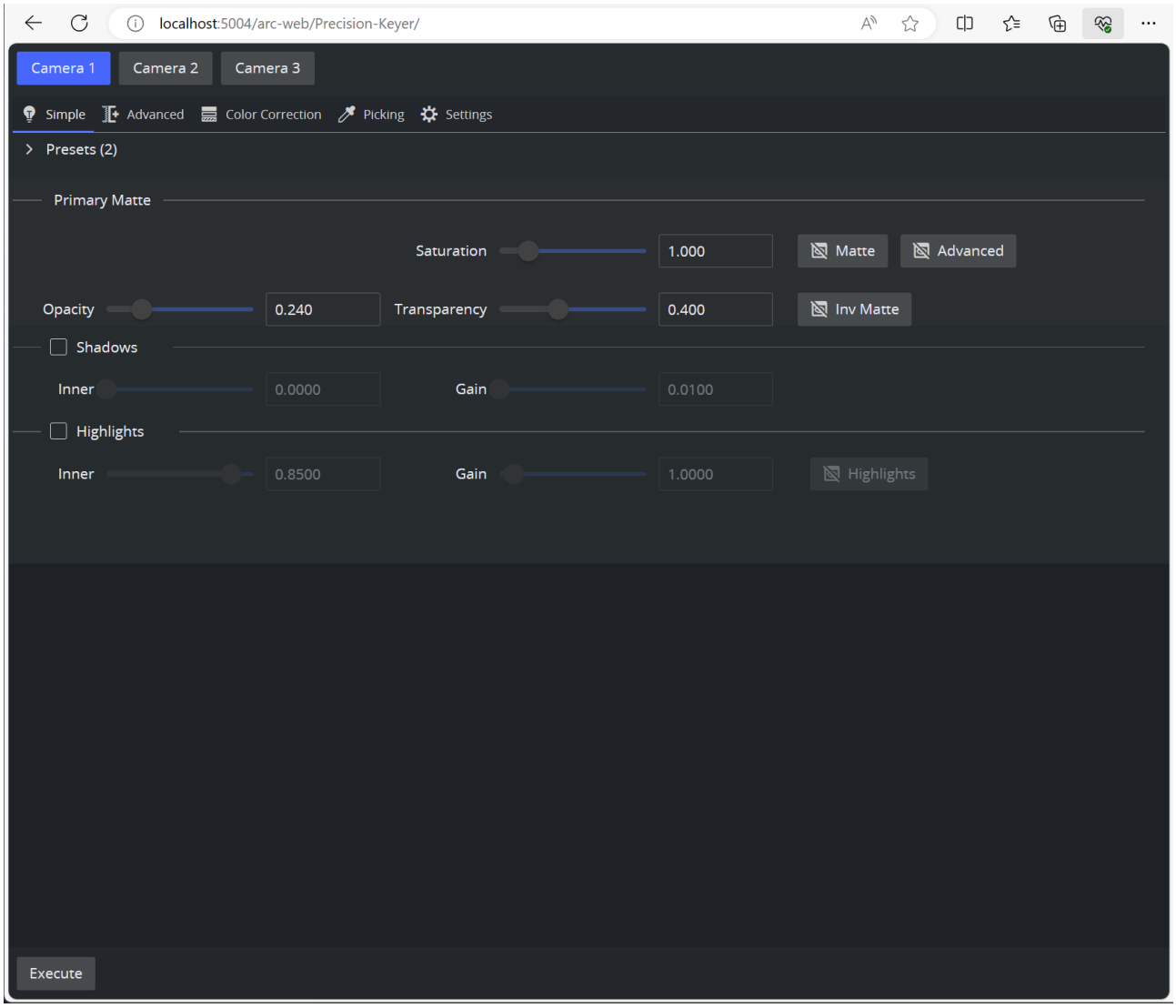
The entire UI can be visualized in an external web browser. Top open it with the default browser go to **View > Web Views > Precision Keyer**.

Copyright © 2025 Vizrt



Or enter the URL *http://localhost:5004/arc-web/Precision-Keyer/* into a web browser. Replace *localhost* with your hostname to reach the page from an external machine.

Copyright © 2025 Vizrt

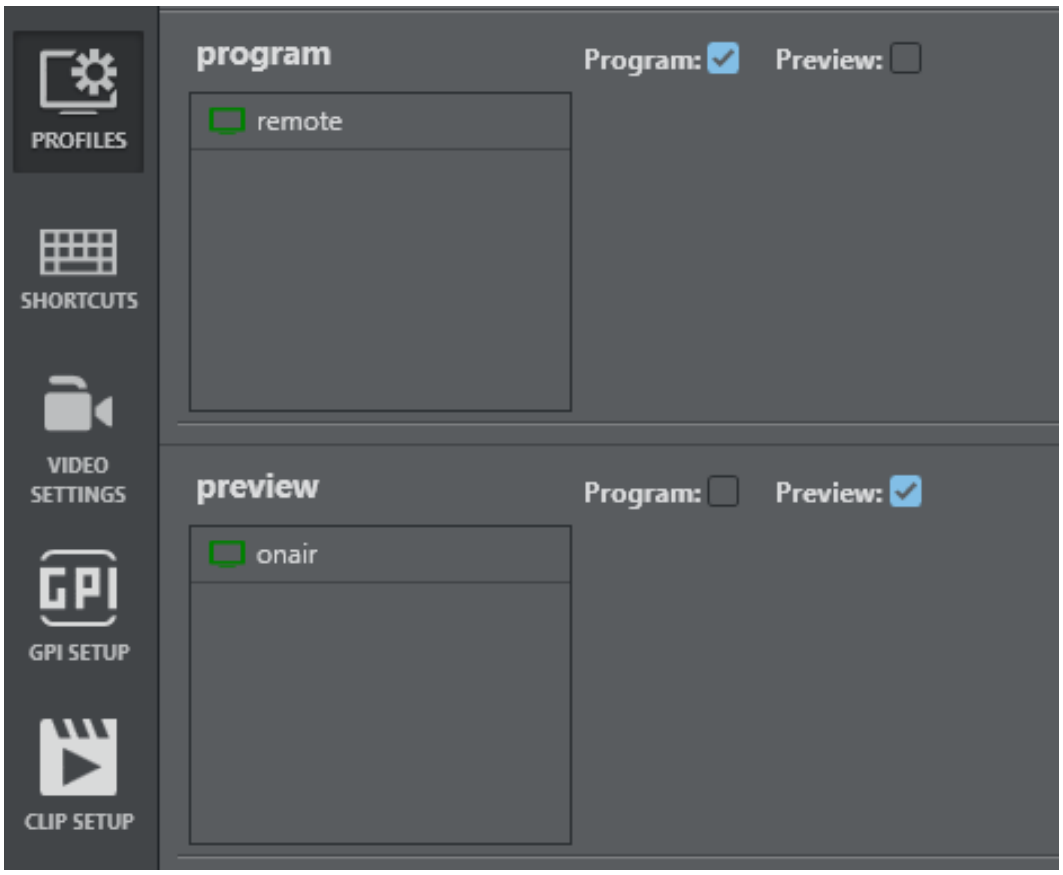


Copyright © 2025 Vizrt

5.19 Program and Preview

If a preview channel is configured in the active profile, you can execute any action on the preview channel:

- Previewing by Keyboard Shortcut
- Previewing by CTRL Modifier



A sample profile configuration defining a program and preview channel

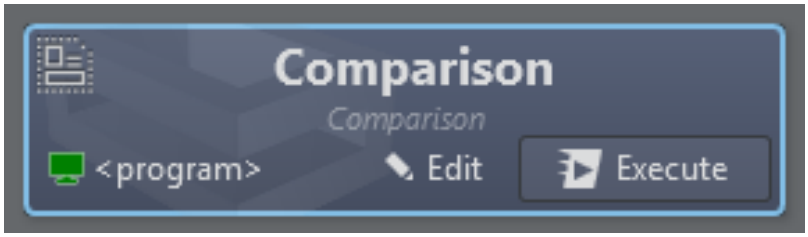
5.19.1 Previewing by Keyboard Shortcut

Select one or multiple actions and use the keyboard shortcut defined in the **Shortcut** section (default **F9**).

5.19.2 Previewing by CTRL Modifier

Press and hold the **CTRL** key and execute an action by clicking the **Execute** button. The action is executed on the preview channel instead of the channel defined in the action.

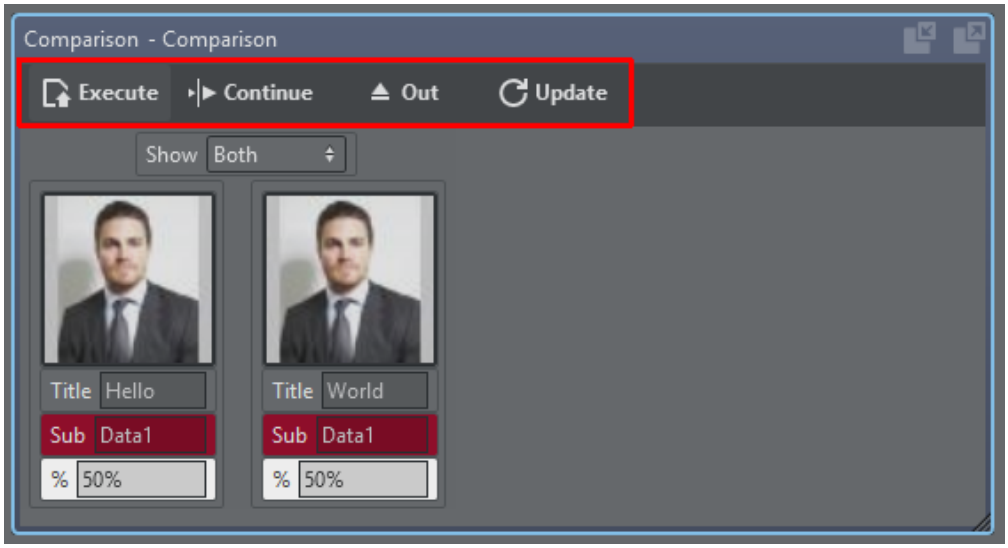
Copyright © 2025 Vizrt



Either press **F9** or press and hold **CTRL** and left click the **Execute** button to execute the action on the preview channel.

5.19.3 Templates

In case of a template action, the **CTRL** modifier is extended to the entire command header:



When holding the **CTRL** key and hitting any of the header buttons, the respective commands are sent to the preview channel.

Copyright © 2025 Vizrt

5.20 Projects

In Viz Arc, projects consist of two main elements:

- **Scenes:** The Viz Scene or Scenes the project controls.
- **Actions:** Buttons that control different parameters of any Viz Scene within a project when activated.

Optionally, a **Project** may also include:

- **Custom Script:** For more dedicated logic and controls.
- **Virtual Studio or Augmented Reality** objects configuration.

Projects can be managed (Open, Create, Save, etc.) from the Main menu or Project toolbar.

> **Note:** All projects are stored in the Projects folder, see Files and Folders.

Copyright © 2025 Vizrt

5.21 Templates

- Creating a New Template
- Using a Template
    - Global Template Properties
    - Template Controls
- Adding Templates to a Master Scene
- Using Templates on a Scene Containing Multiple Top Level Control Objects
- Unreal Templates
    - Creating a New Unreal Template
    - Unreal-specific Features
    - Using a Template
- Flowics
    - Preparing Flowics Graphics for Use in Viz Arc
    - Creating a New Flowics Template
    - Global Data Provider - Custom Data
    - The logic of Execute and Update
    - Executing Custom Commands

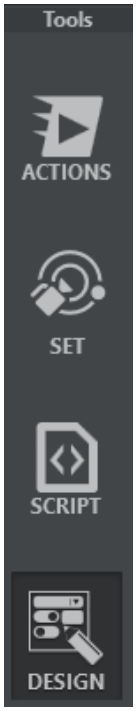
Viz Arc lets you create custom templates for both Viz Engine scenes and Unreal Engine levels. The templates are fully customizable and a script can be used for even more flexibility, including data integration and custom logic.

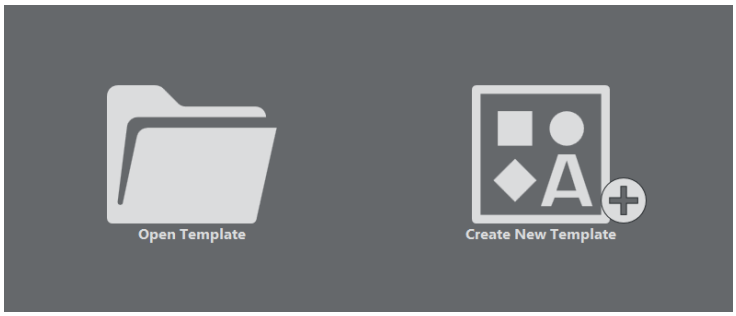
> ⚠ **Note:** Before creating a new template, you must ensure that Graphic Hub has the required Add-Ons to store the templates, see Configuring Graphic Hub.

5.21.1 Creating a New Template

- Under the **Tools** section, select **DESIGN**.
- Next, click **Create New Template**.

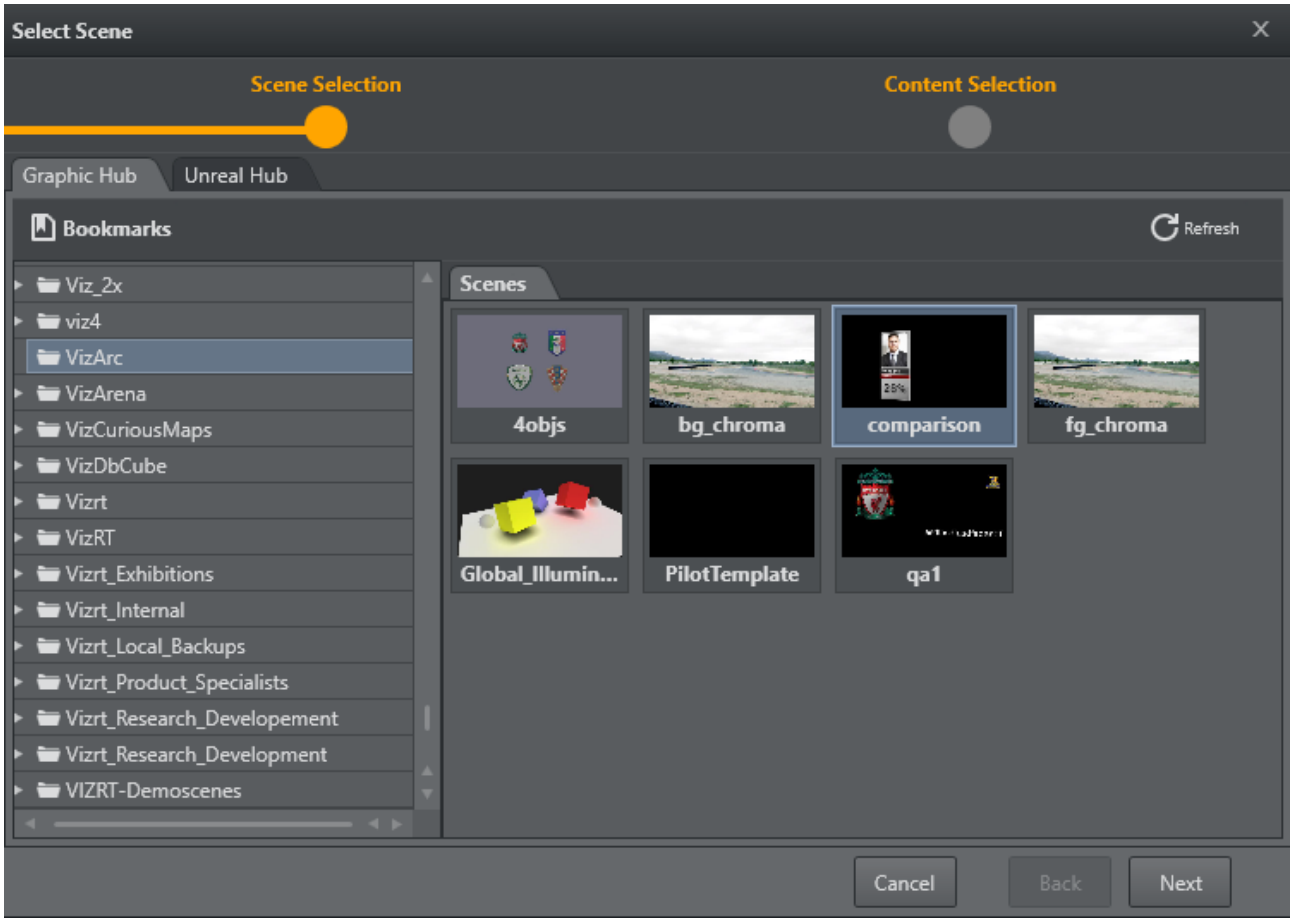
Copyright © 2025 Vizrt





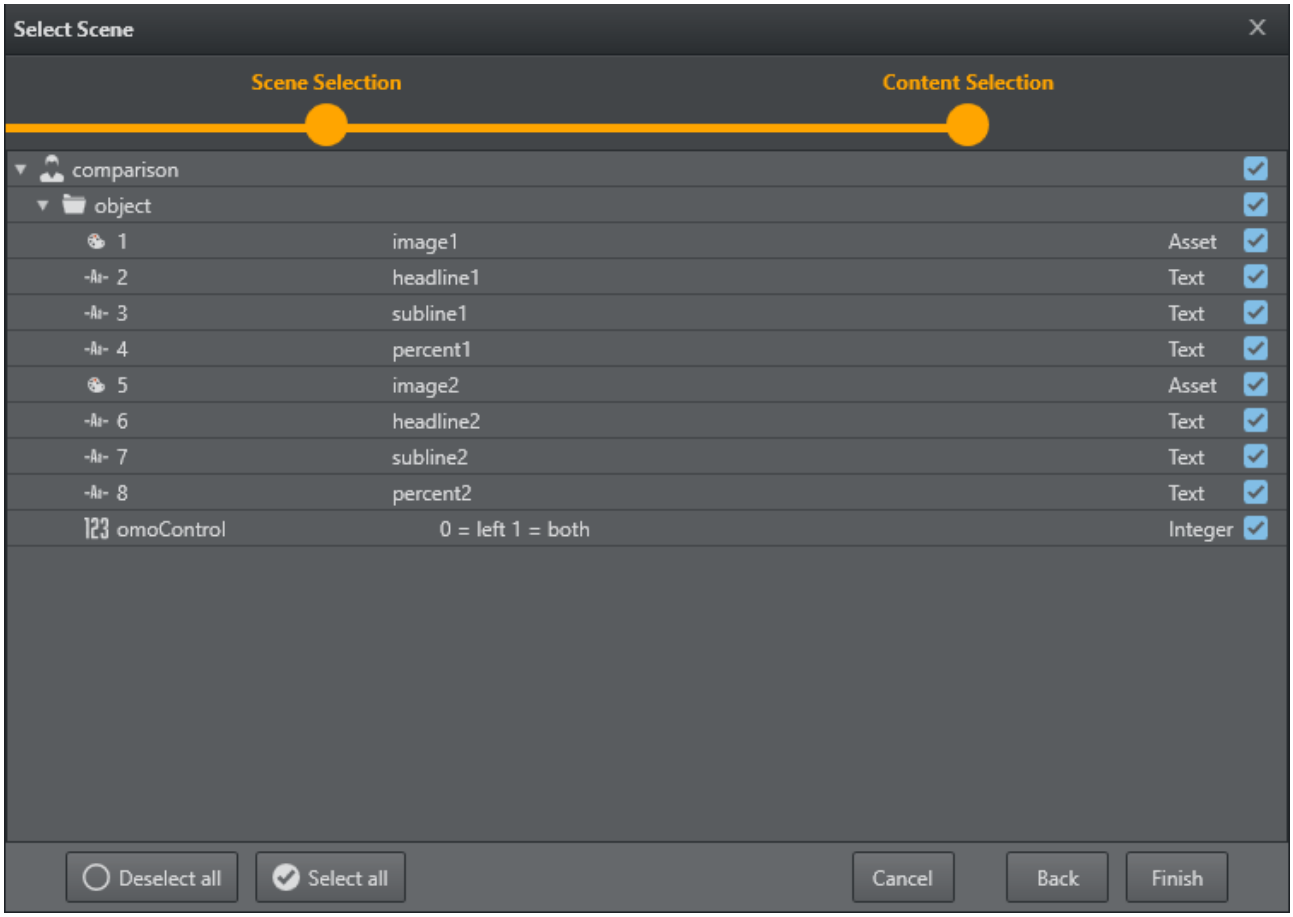
A browser window appears where you can select either a Viz or an Unreal scene:

Copyright © 2025 Vizrt

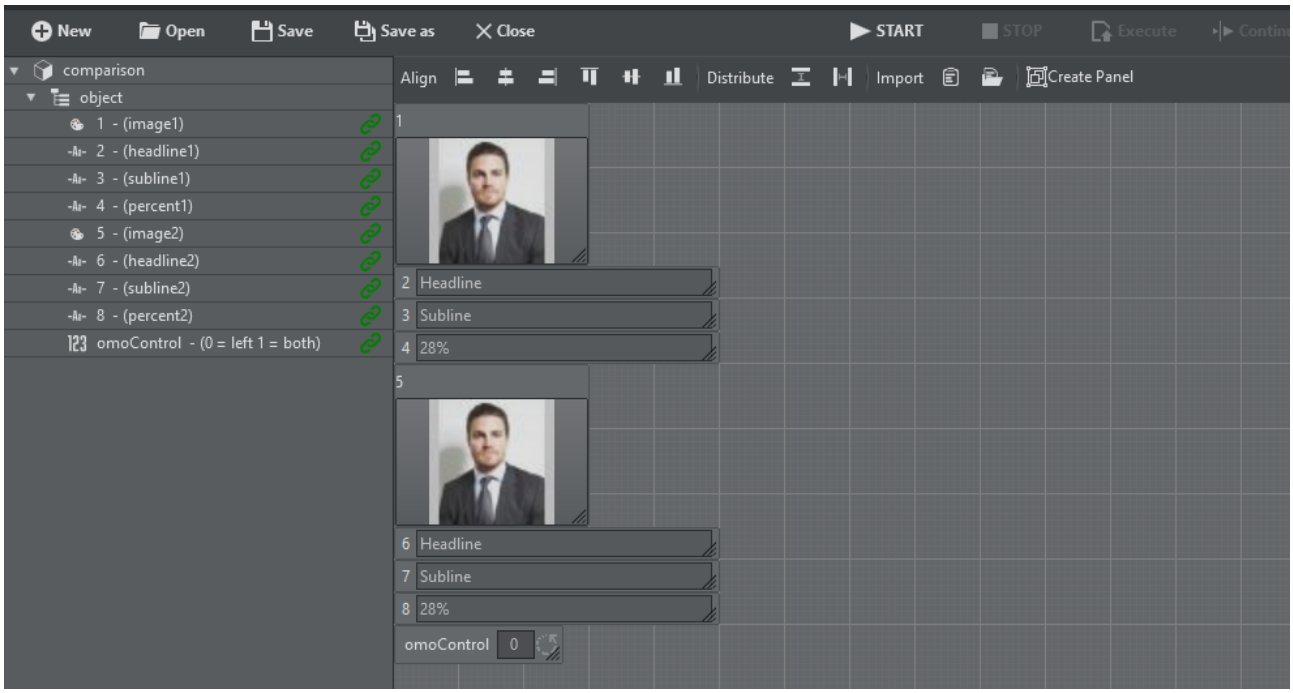


Click **Next** to view Control Objects found within the scene. Select the Control Objects for which a default UI is to be created:

Copyright © 2025 Vizrt



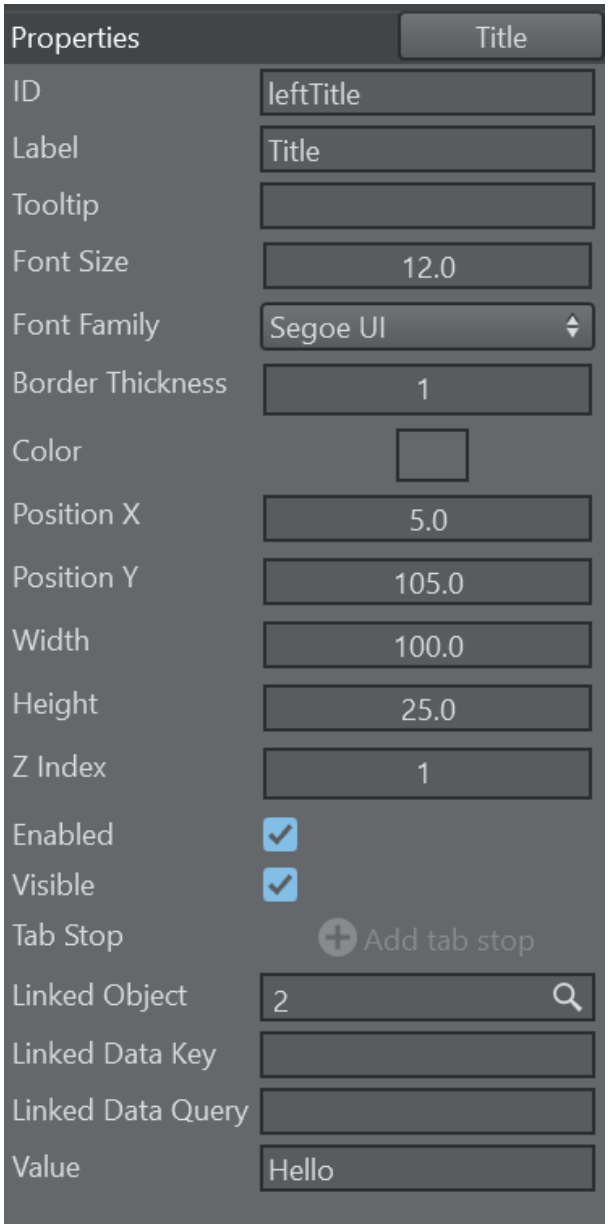
Click **Finish**. A default UI based on the selected Control Objects has now been created:



Copyright © 2025 Vizrt

The left hand side contains a list of the Control Objects' IDs along with their description. A **green link icon** indicates whether the Control Object is linked to a UI element.

A property panel on the right shows editable properties of the selected UI element, or, in case of multi-selection, the properties that are common to the selection. A sample of a text parameter property:



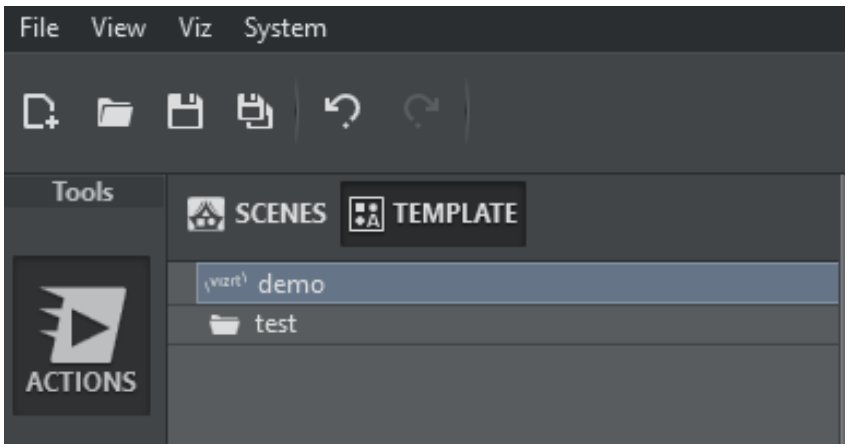
- **ID:** The unique internal ID of the parameter that can be used in the script code to reference the parameter.
- **Label:** The label of the parameter.
- **ToolTip:** An optional ToolTip.
- **Font Size:** The size of the font used in the textual parts of the component.
- **Font Family:** The font used in the textual parts of the component.
- **Border Thickness:** The border thickness of the parameter.
- **Color:** The background color of the parameter.
- **Position X:** The horizontal pixel position of the upper left corner of the parameter.

Copyright © 2025 Vizrt

- **Position Y:** The vertical pixel position of the upper left corner of the parameter.
- **Width:** The width of the parameter.
- **Height:** The height of the parameter.
- **Z Index:** The z sort index of the parameter. In case of overlap with other parameters this value can determine whether the parameter is in front or behind the overlapping component.
- **Enabled:** Whether the parameter is usable for user interaction.
- **Visible:** Whether the parameter is visible to the user.
- **Tab Stop:** The order of the tab-stop.
- **Linked Object:** In case the parameter is linked to a ControlObject ID or a Blueprint variable.
- **Linked Data Key:** The name of the DataMap key (see DataMap Linking).
- **Linked Data Query:** The query for the DataMap value (see DataMap Linking). Can be empty for simple data mapping.
- **Value:** The actual value of the parameter.

5.21.2 Using a Template

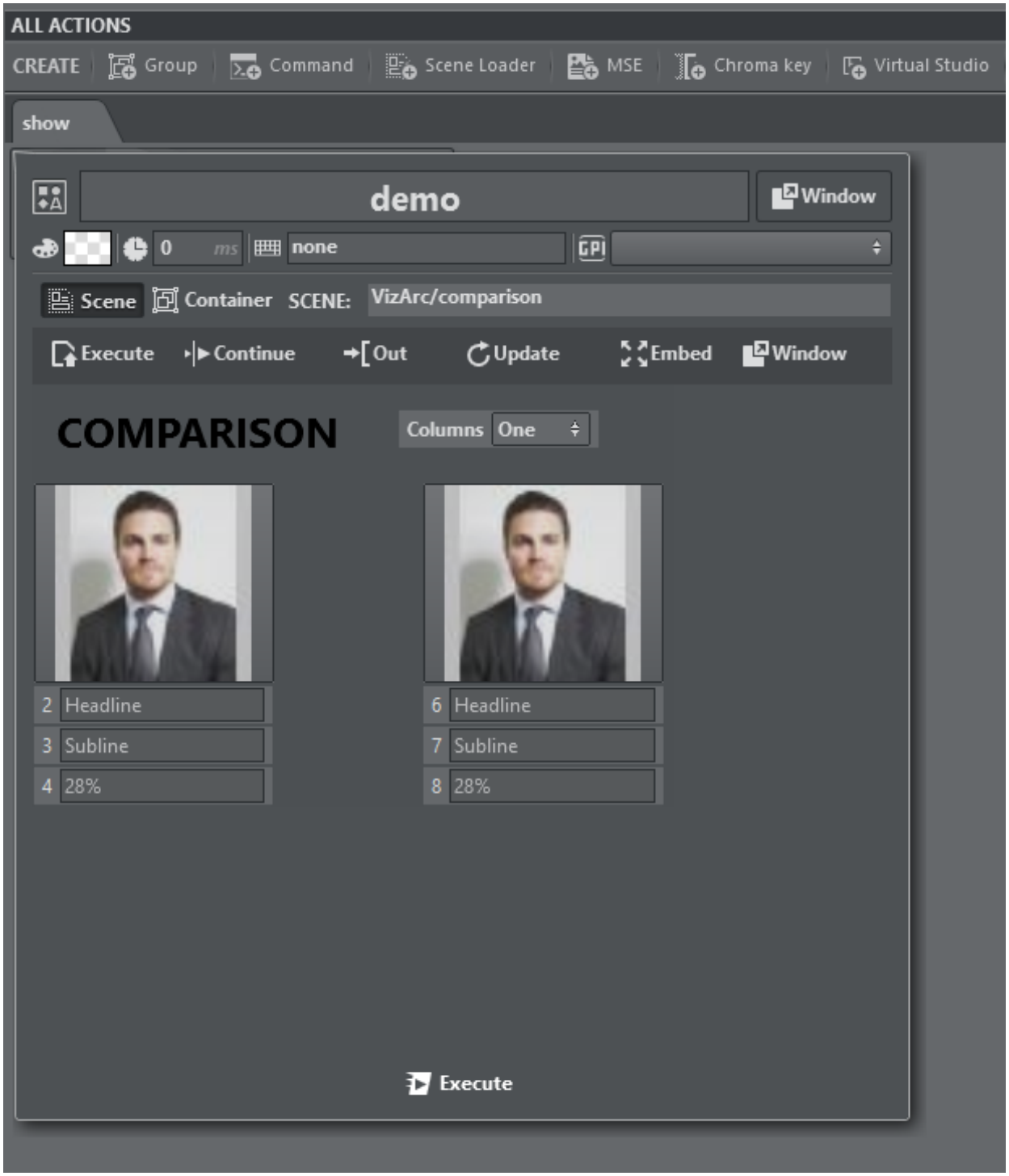
Templates are available in the action panel.

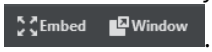


The list of templates can be refreshed by right-clicking on the template list canvas and selecting **Refresh** from the context menu.

To create an instance of a template, drag and drop it to the action canvas.

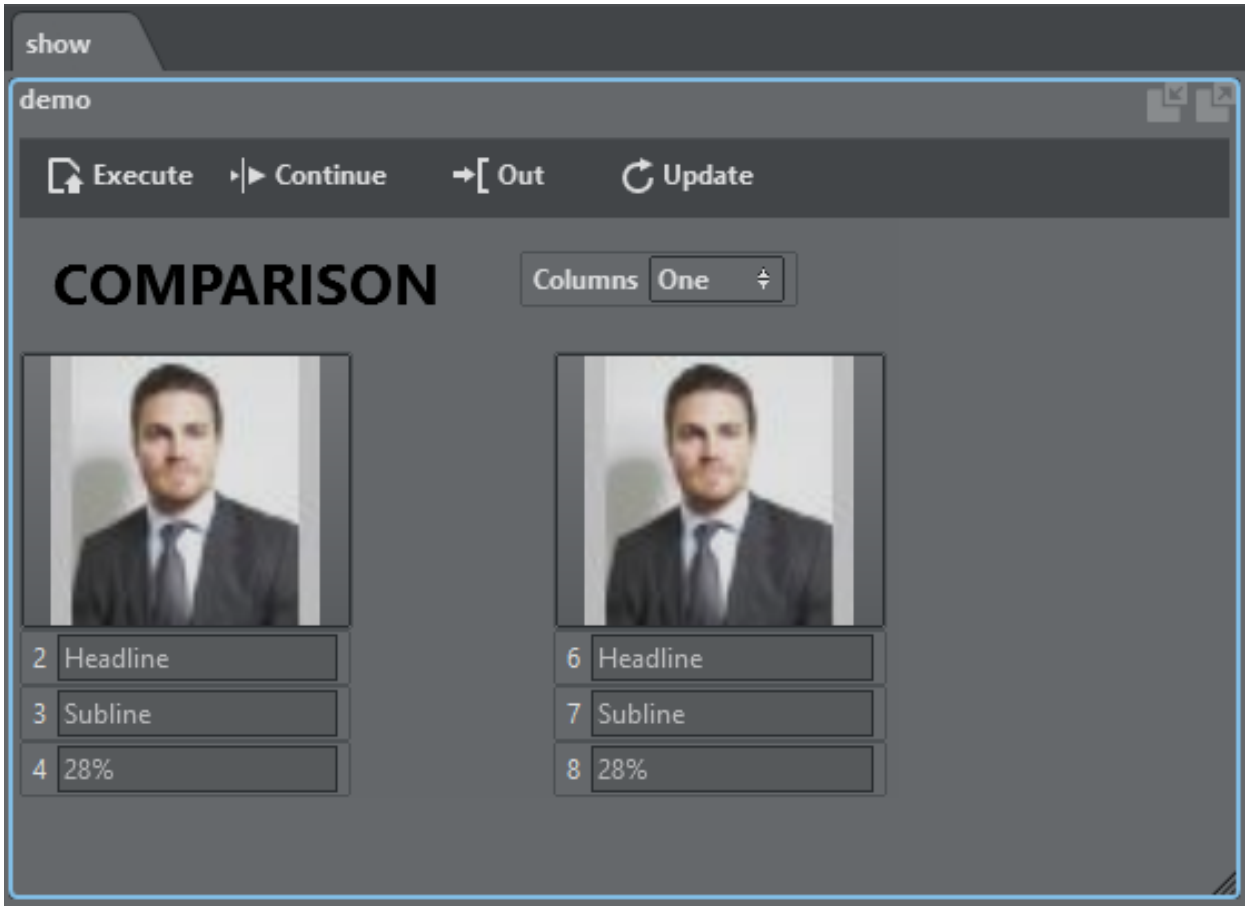
Copyright © 2025 Vizrt



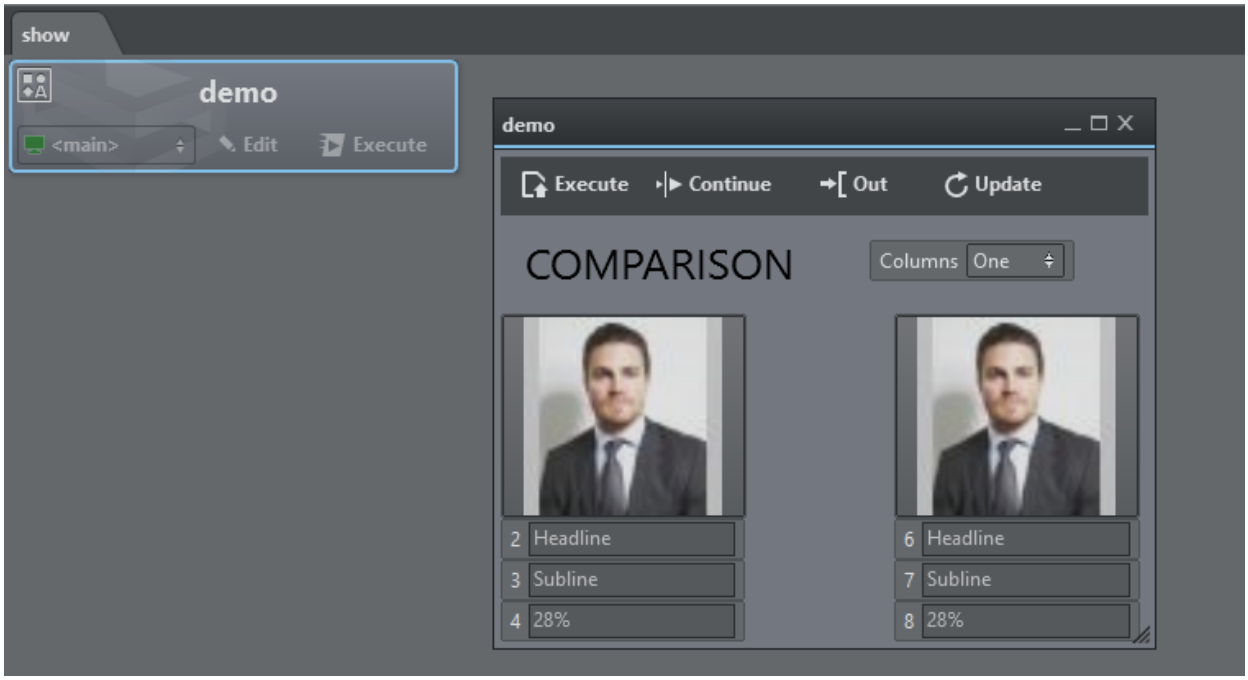
The template action can be opened in the same way as all other actions in a popup window. It can detached as a window that always stays on top, or embedded into the action canvas using the **Embed** or **Window** buttons 

.

Copyright © 2025 Vizrt



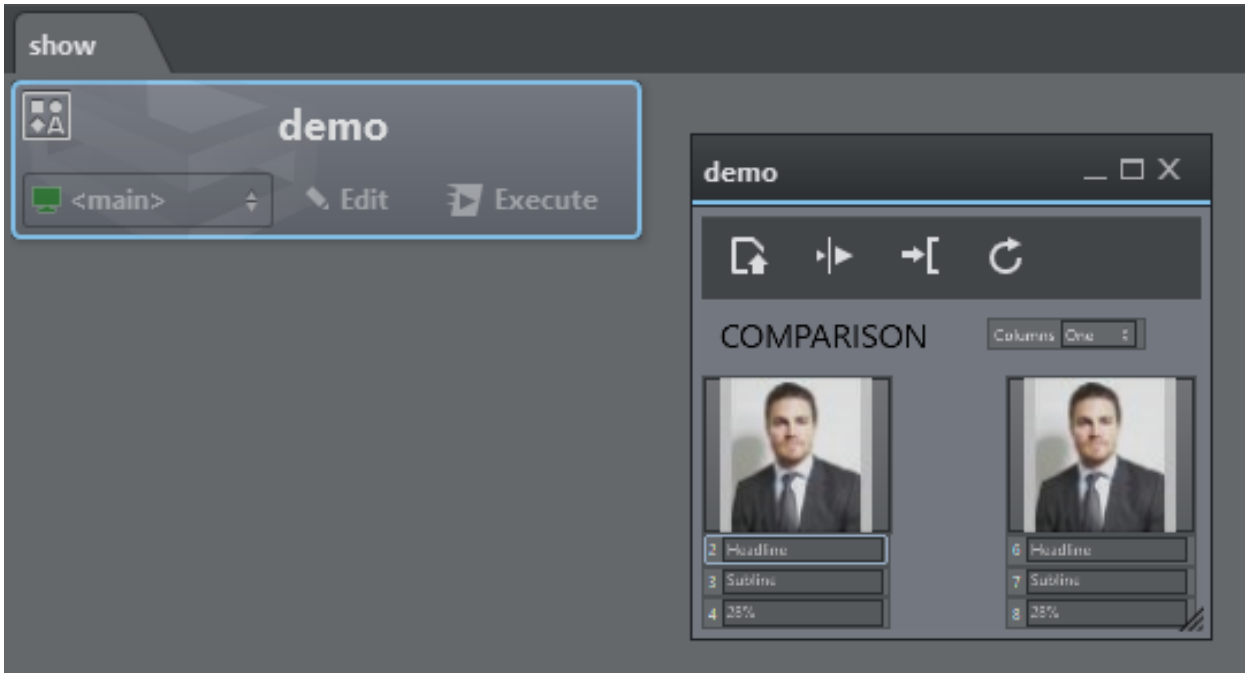
Template action as an embedded window is shown above. The embedded window can be resized by percentage or by using the resize grip in the bottom right corner of the action.



Copyright © 2025 Vizrt

Template action as a detached window is shown above.

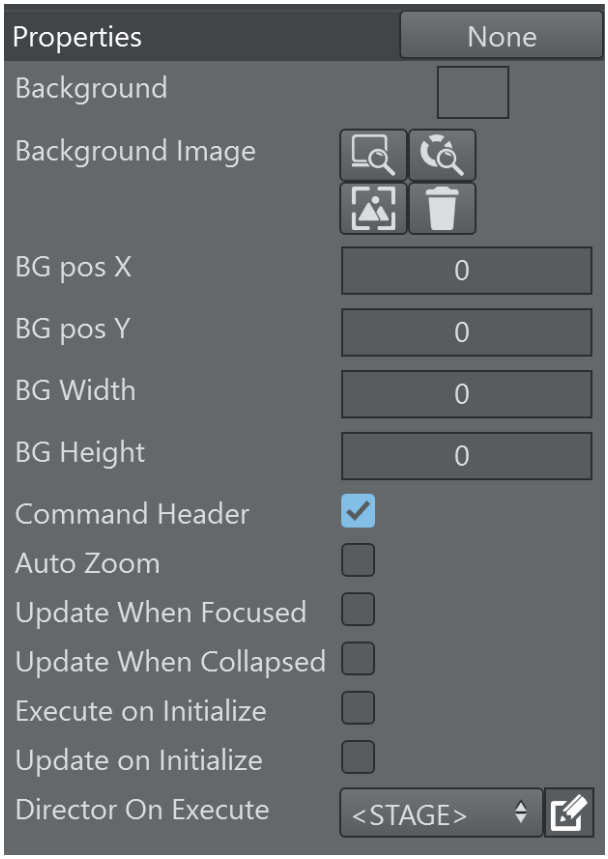
You can zoom in and out of each template action using the shortcuts **CTRL +** to increase size, **CTRL -** to reduce size, and **CTRL + 0** to restore the original size.



A zoomed out version of a template action window is shown above.

Global Template Properties

When creating or modifying a template and when in the UI editing mode, click on the background canvas to access a template's global properties:



- **Background:** Sets a default background color. When dragging the template onto the action canvas, the action color is the background color.
- **Background Image:** Lets you choose a background image for the template either from a local file system, the GH or the Media Service. The snap button grabs the current frame of the editing Engine.
- **BG pos X:** The horizontal pixel position of the background image.
- **BG pos Y:** The vertical pixel position of the background image.
- **BG Width:** The pixel width of the background image.
- **BG Height:** The pixel height of the background image.
- **Command Header:** Whether the template displays the default command header (Execute, Continue, Update, Out).
- **Update When Focused:** When enabled, the template receives callbacks only when the template is selected, has focus, and is opened in either embedded or window mode. Relevant callbacks are: OnTimer, OnStreamDeckKey, OnDataMapValueChanged. When disabled, the template receives those callbacks when instantiated on the action canvas in any form (embedded, windowed or closed and without being explicitly selected).
- **Update When Collapsed:** When enabled, the template can receive callbacks when the template is collapsed as action button. Relevant callbacks are: OnTimer, OnStreamDeckKey, OnDataMapValueChanged. When used in combination with the flag **Update When Focused** unchecked, the template always receives the relevant callbacks, when not selected and collapsed to a standard action button.
- **Execute on Initialize:** Executes the template when the global Initialize button is triggered.
- **Update on Initialize:** Updates the template when the global initialize button is triggered.

Copyright © 2025 Vizrt

- **Director On Execute:** This dropdown menu contains the list of directors of the scene. Select a director that should be triggered when executing and continuing the template. By default the entire Stage is animated when <*STAGE*> is selected.

Template Controls



Execute

A template action can be triggered using the **Execute** button above the template or through the system keyboard shortcut **Execute Selection**. When the action is collapsed, the regular **Execute** button on the action does the triggering. When executing, the scene that is associated with the template is loaded, the Control Objects are being updated and the stage is being played. In case an engine of the associated channel has a **Concept** configured, the scene in that concept is loaded.

Continue

The **Continue** button or the **Continue Selection** keyboard shortcut sends a continue command to the scene stage. When the action is collapsed, the **Execute** button executes a **Continue** when clicked with the **right mouse button**.

Out

The **Out** button or the **Out Selection** keyboard shortcut removes the scene associated to the template from the renderer, thereby causing a **hard out**.

Update

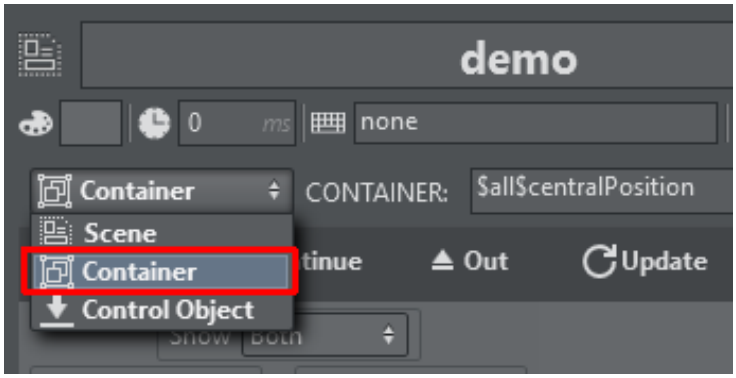
The **Update** button or the **Update Selection** keyboard shortcut sends the currently set Control Objects to the renderer and update the values accordingly. This can be useful for updating live data while a scene is on-air.

> **Info:** The system **Execute Selection**, **Continue Selection**, **Out Selection** and **Update Selection** shortcuts only work when a single action is selected.

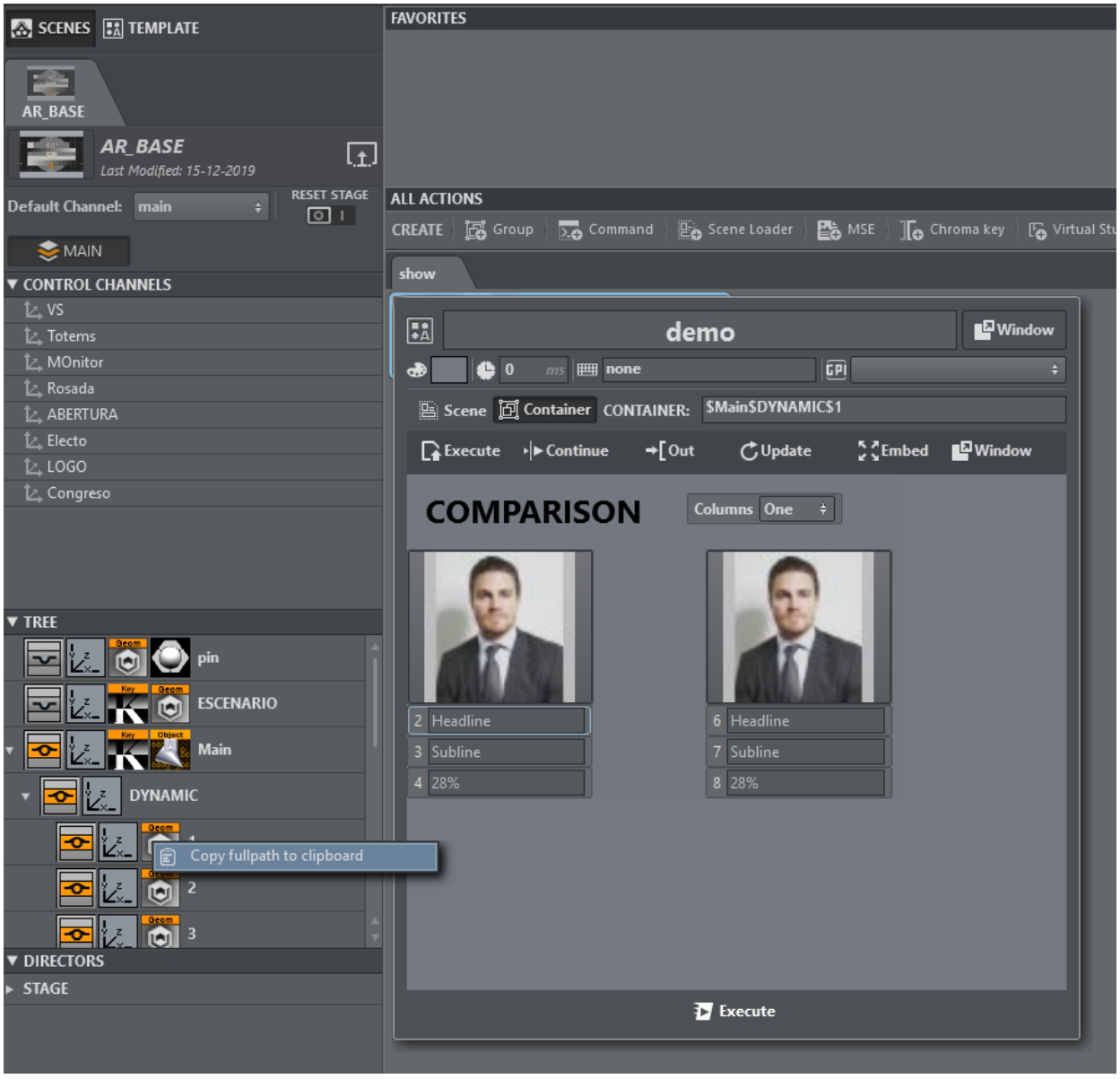
5.21.3 Adding Templates to a Master Scene

By default, a template action is triggering a stand-alone scene and is thus useful for simple lower third graphics, tables etc. When editing the template action, it's possible to toggle the destination of the template's action to a container path of another scene.

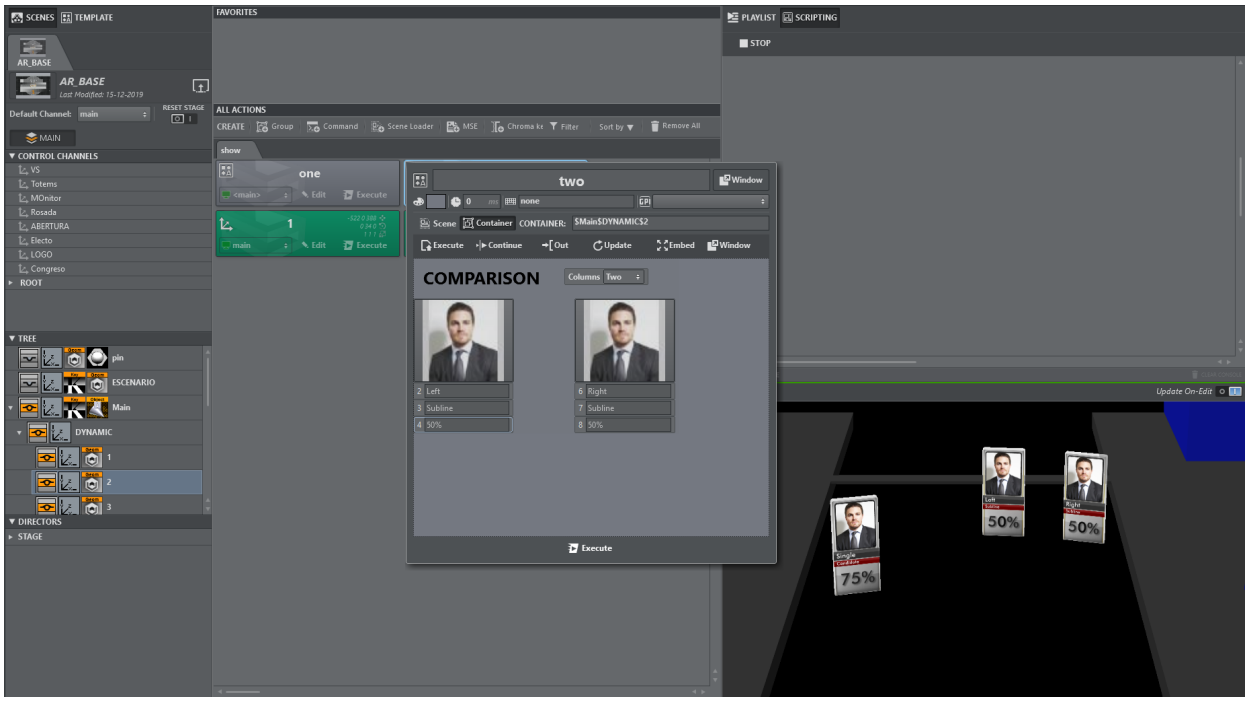
Copyright © 2025 Vizrt



The container path can be entered manually or can be obtained from the scene tree by right clicking the desired container and selecting **Copy full path to clipboard**.

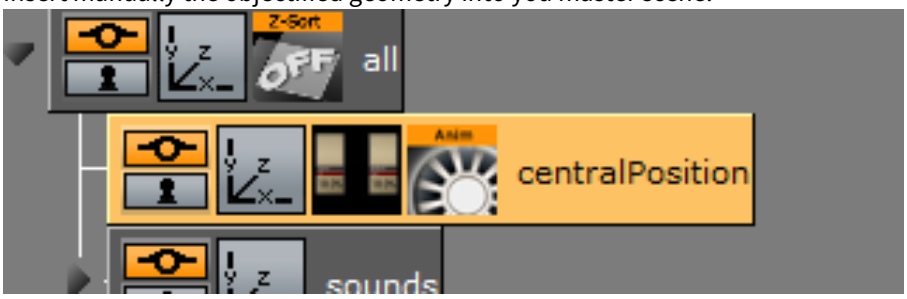


Executing those actions results in the objectified object being loaded into the destination container path and the respective Control Objects being updated.



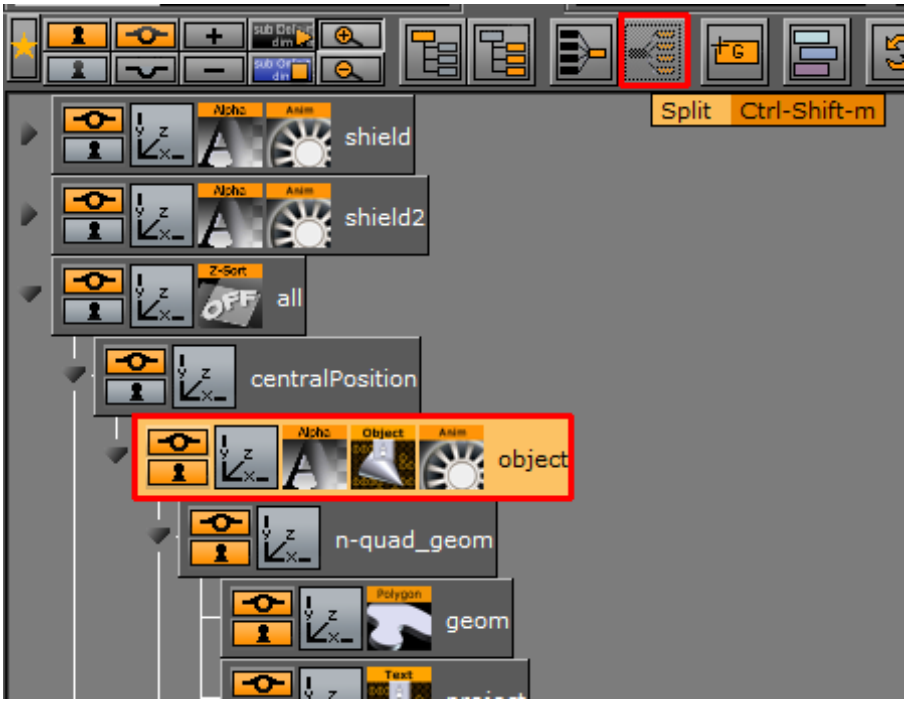
Another way to insert template scenes into a master scene is the following:

- Create your template based on a separate scene.
- Insert manually the objectified geometry into you master scene.



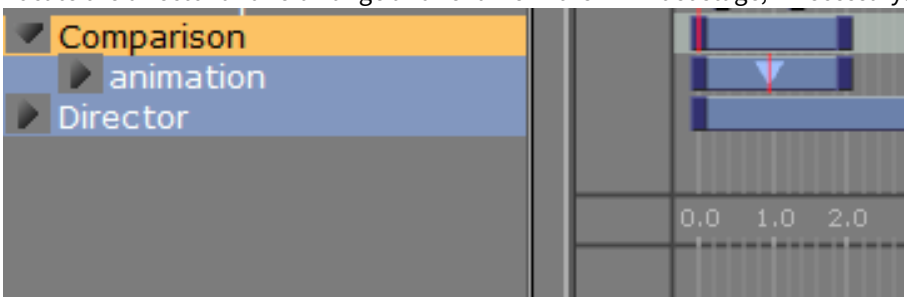
- Split the merged geometry.

Copyright © 2025 Vizrt

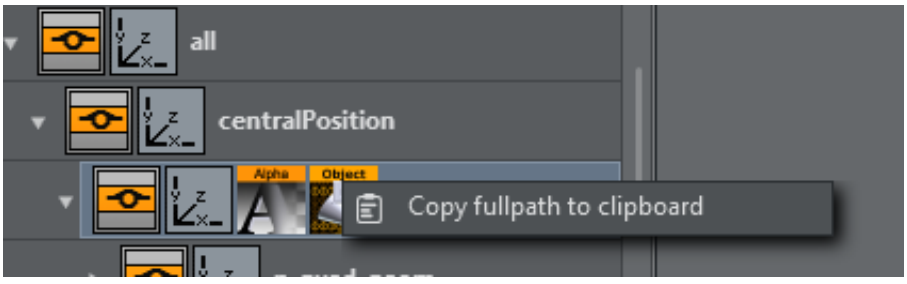


You'll obtain a container containing your top ControlObject as in the stand-alone scene.

- Locate the director and re-arrange and rename in the Viz Artist stage, if necessary.



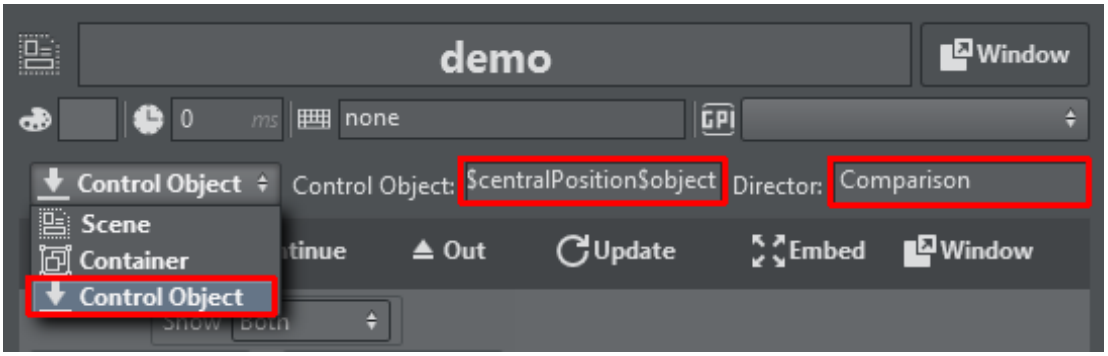
- Save the scene and reload it in Viz Arc.



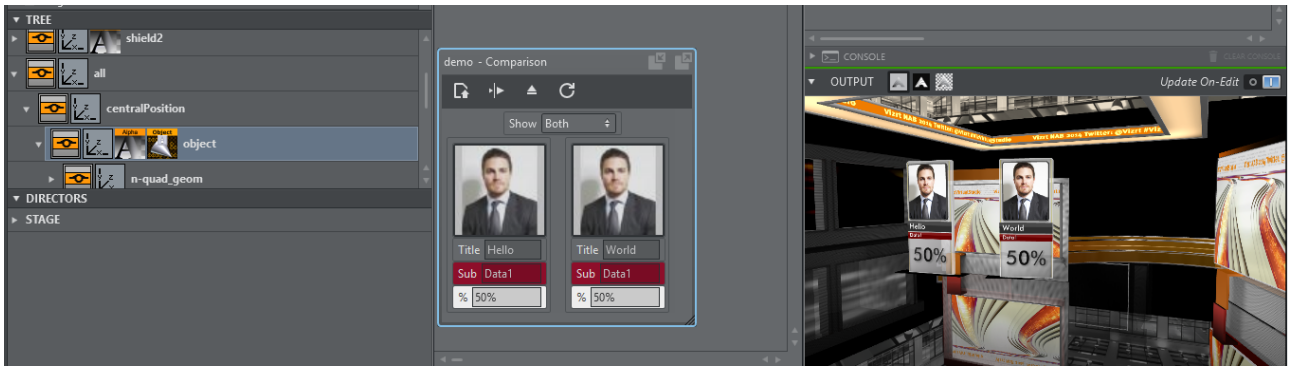
Copy the full path of the container containing the **Control Object**.

- In the template select **Control Object**, paste the container path to *Control Object* and the path to its Director to *Director*.

Copyright © 2025 Vizrt

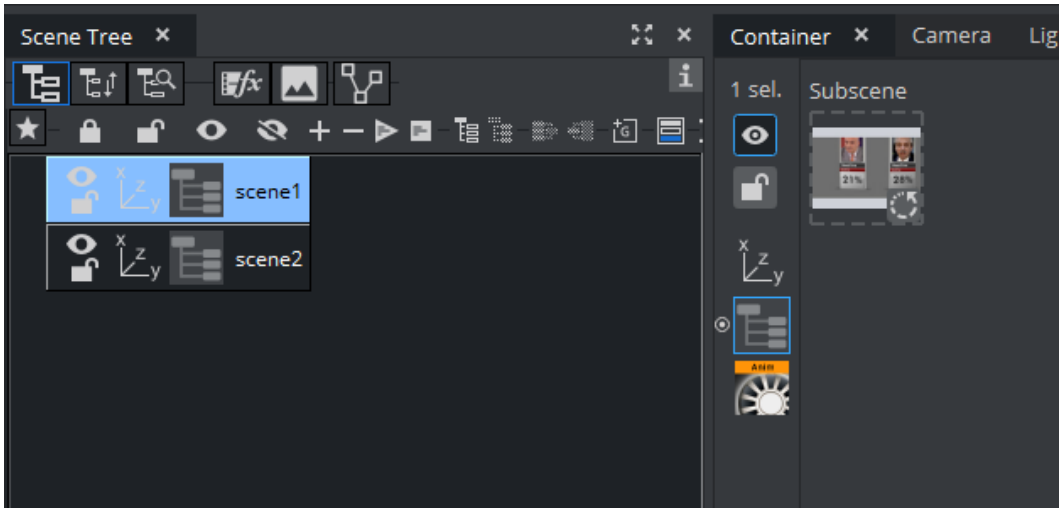


You'll be able to operate your template with execute/continue, while the scene is embedded in the AR/VR master scene.



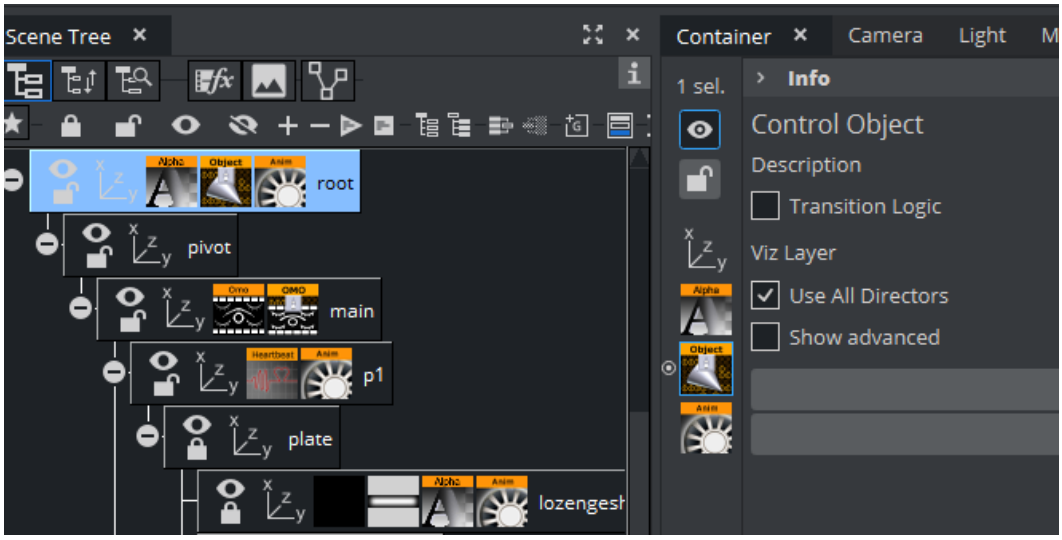
- In case a scene is loaded in the scene tree as a **SUBSCENE**, manually append the string ***SUBSCENE*TREE*$PathToControlObject**.

As an example, consider a scene with a container *$scene1*.



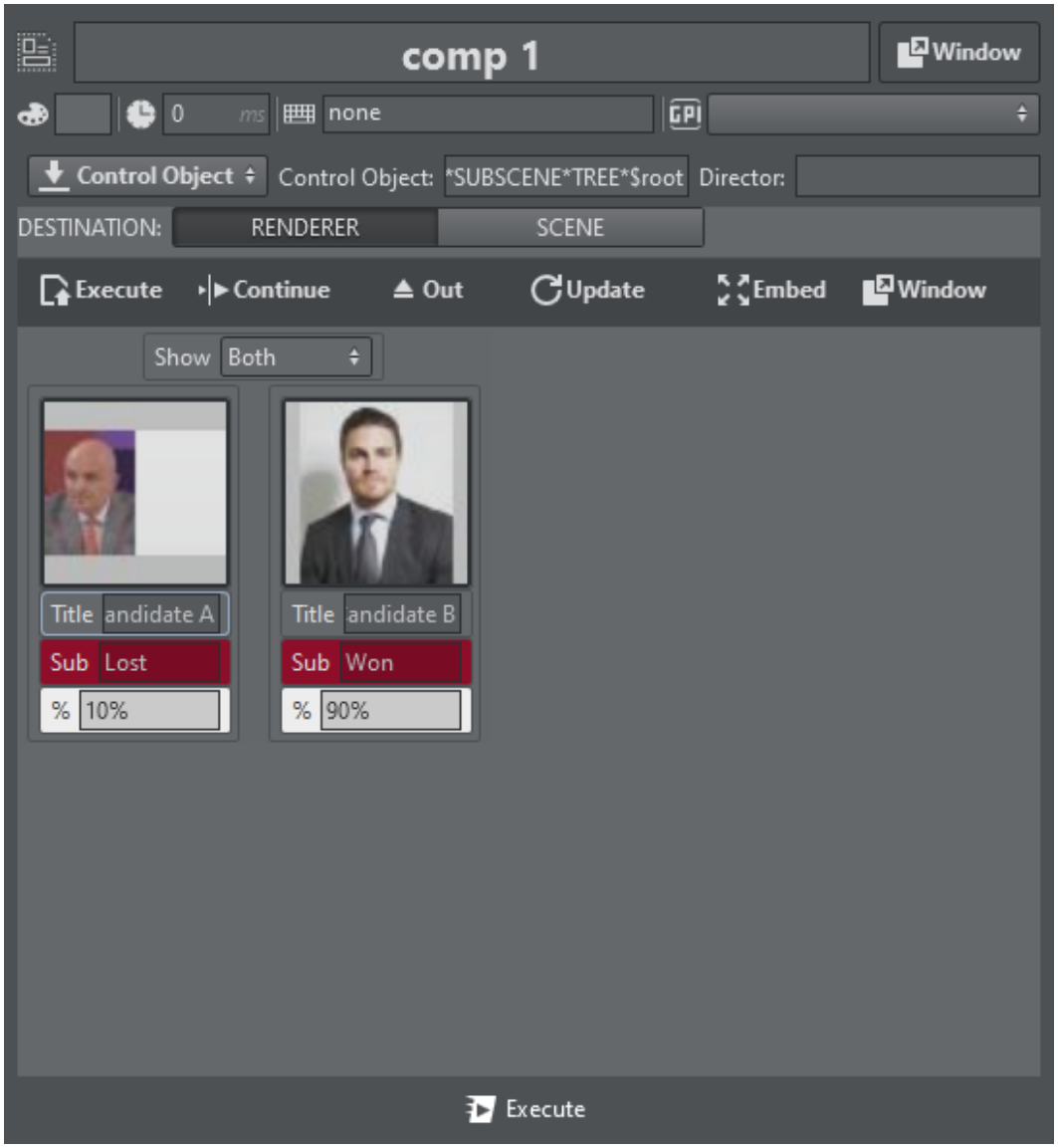
This container contains a **SUBSCENE**, on the subscene another scene is loaded that has a ControlObject on the *$root* container.

Copyright © 2025 Vizrt



A template can now be used to control the controls of the subscene using the path **$scene1*SUBSCENE*TREE*$root** in the *Control Object* input field.

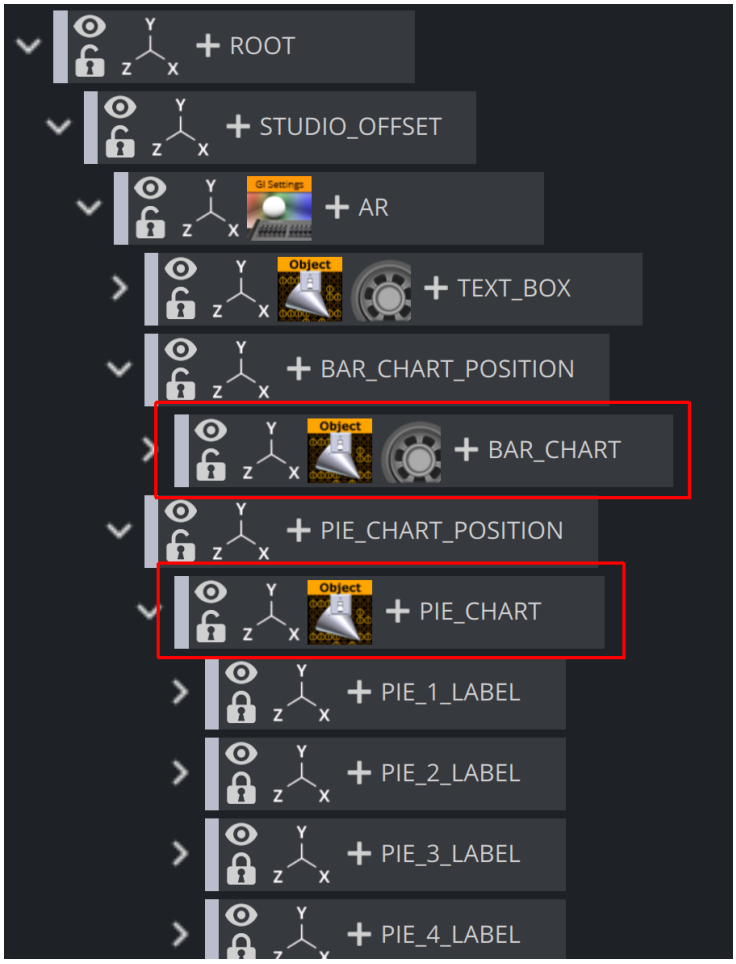
Copyright © 2025 Vizrt



5.21.4 Using Templates on a Scene Containing Multiple Top Level Control Objects

Within a large AR scene, you might want to create subtrees separating different graphics elements. For example, a subtree containing a bar chart and another subtree containing a pie chart graphics.

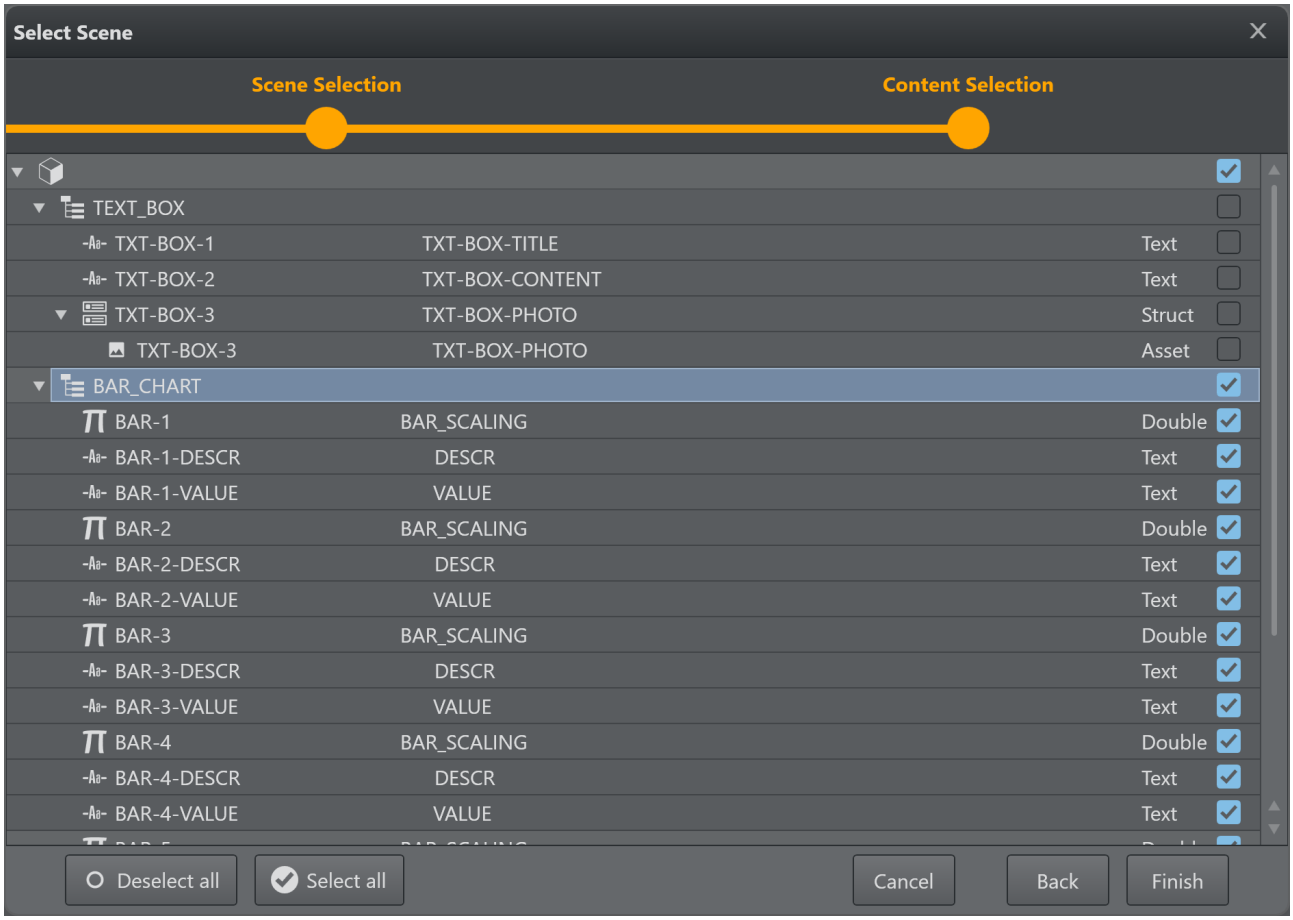
Copyright © 2025 Vizrt



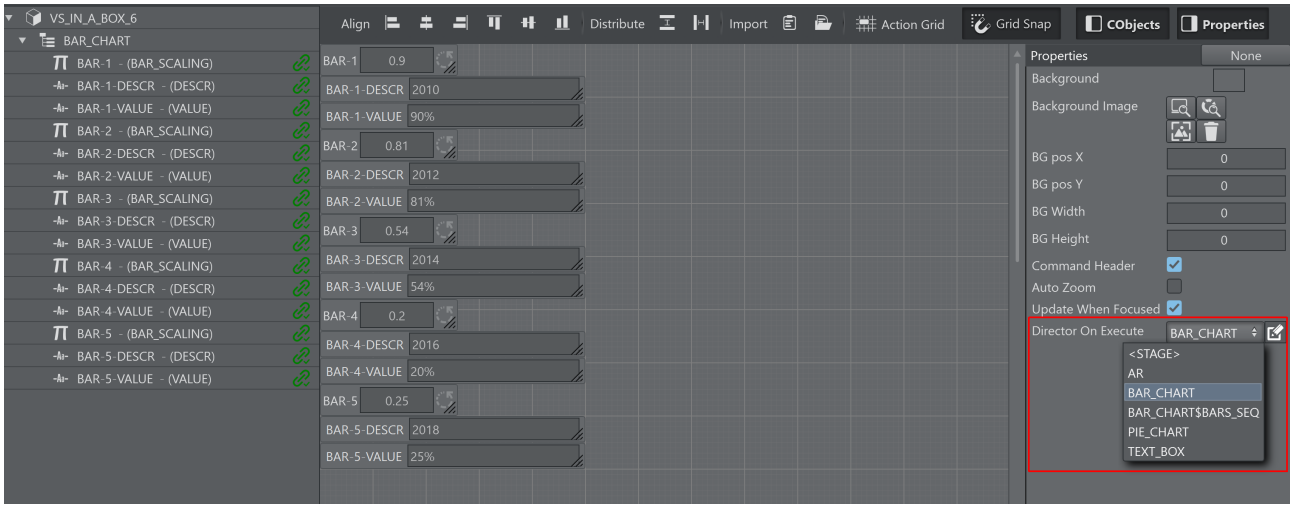
When using this kind of tree structure having Control Objects spread around the scene as sibling nodes, it is possible to directly create templates for each individual top level Control Object.

While creating a new template by selecting the scene from the Graphic Hub, you'll be able to select either one of the top level controls. You cannot control multiple top level control objects from within a *single* template.

Copyright © 2025 Vizrt

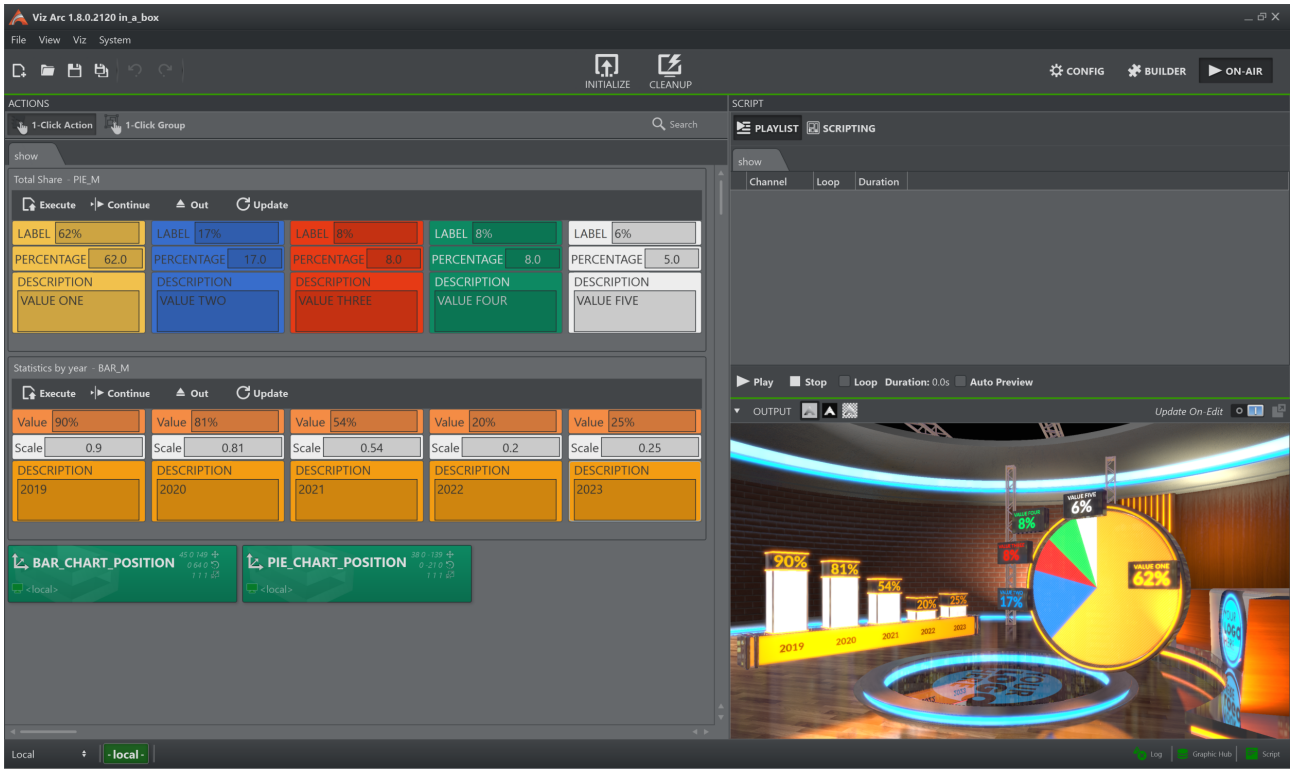


Once a Control Object is selected for the template, choose which director should be used when executing the **Execute** and **Continue** buttons of the template. The selection can be found on the template background property when clicking any empty space on the template canvas. The subscene director has to be built in such a way that there is a stop point when the graphics are fully animated in and that a continue command animates the graphics out again.



Once the templates are finished they can be uses as embedded templates on the action canvas. The operator can trigger the different elements individually.

Copyright © 2025 Vizrt



5.21.5 Unreal Templates

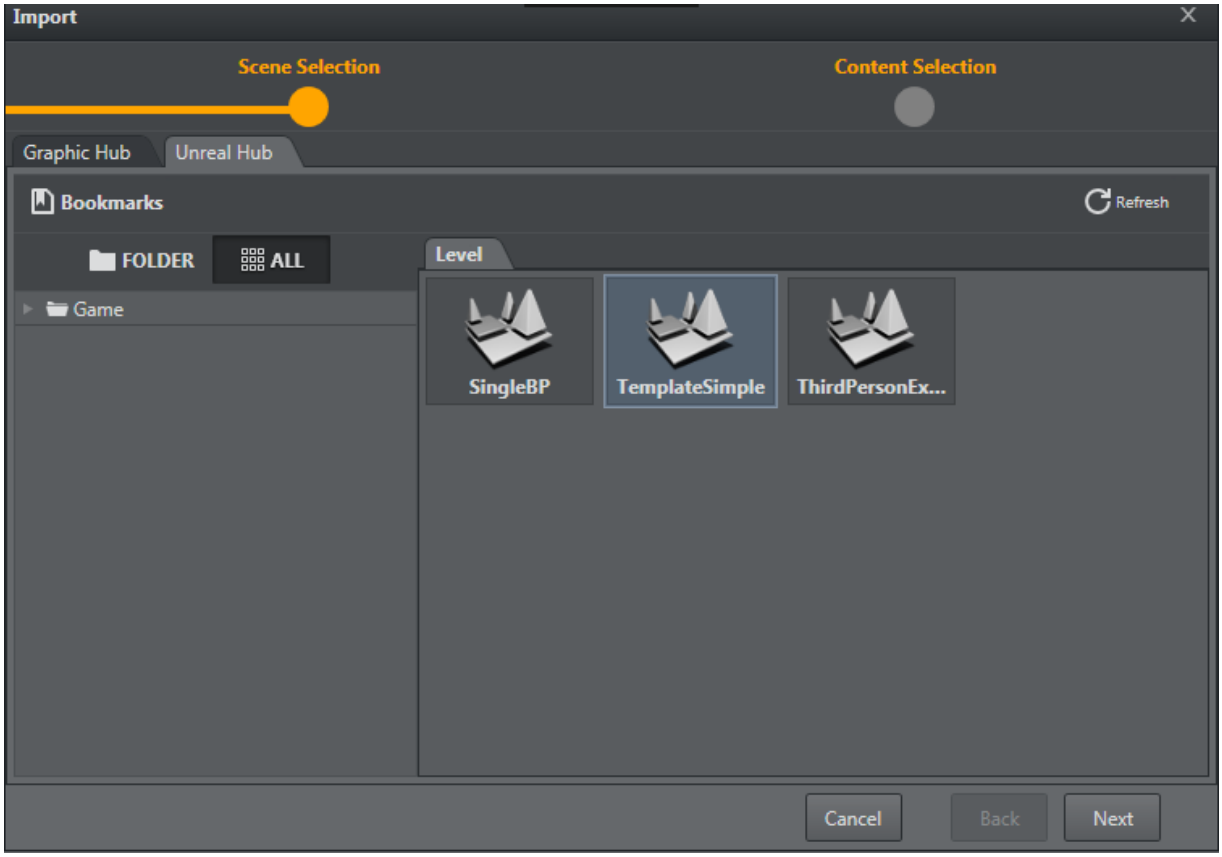
Creating a New Unreal Template

The process of creating an Unreal template is almost the same as that for creating Viz templates, with a few slight differences.

Creating an Unreal Template

- Go to the **DESIGN** section.
- Click **Create new template** to open the wizard.
- Select the **Unreal Hub** tab and select a level.

Copyright © 2025 Vizrt



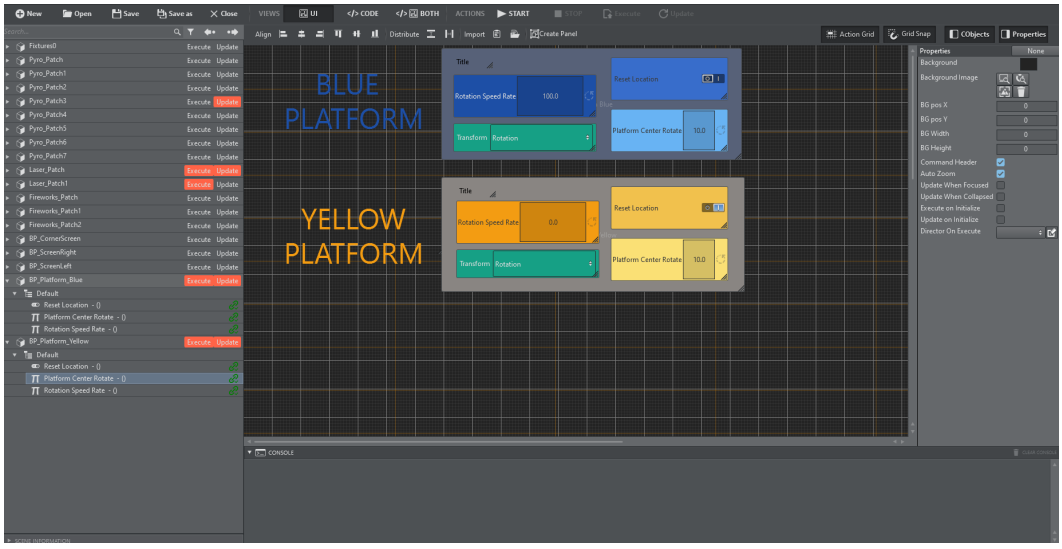
Upon selecting a level and clicking **Next**, the level is set on the configured Unreal editing Engine and all of the blueprint actors and the level blueprint are parsed and displayed on the following screen (as shown below).

The list displays all the blueprint actors found and a list of their categories and exposed objects. You can then select which are to be created when finalizing the template creation.

Copyright © 2025 Vizrt



Unreal-specific Features



The Unreal template in the screenshot above contains the following Unreal-specific features:

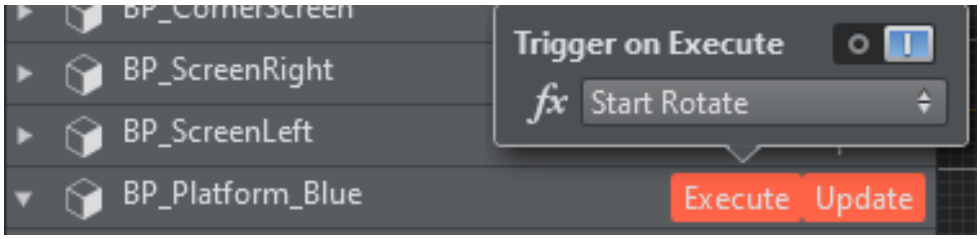
Template Controls

Only **Execute** and **Update** are available for Unreal templates.

Copyright © 2025 Vizrt

Unreal Function Integration

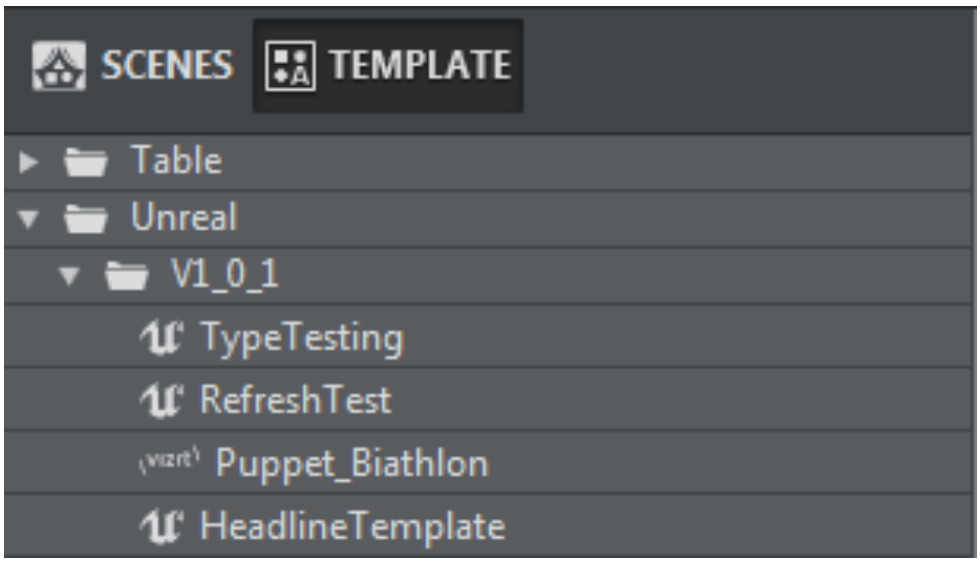
On the left side of the parameter canvas, there is a tree-view containing all the existent blueprint actors. Clicking the **Execute** or **Update** button triggers a popup, allowing you to choose whether the template should call a function and, if so, which one to call (as defined in the Unreal blueprint).



Note: The list of blueprint functions (function drop-down) only displays those functions that have no input parameters.

Using a Template

You can access all saved templates on the action panel. Unreal templates can be identified by the **UE4 icon** that precedes the template name.



Similar to Viz templates, the edit popup for an Unreal template contains settings for loading the level or project on execute.

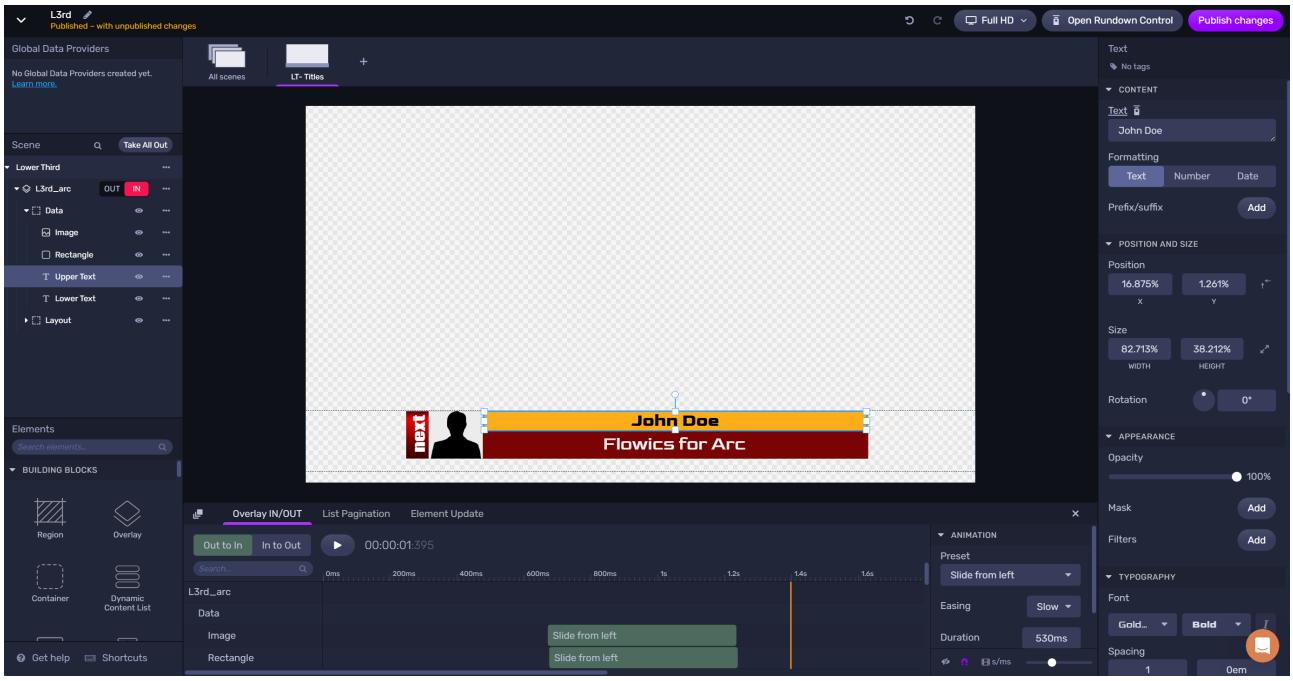
Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

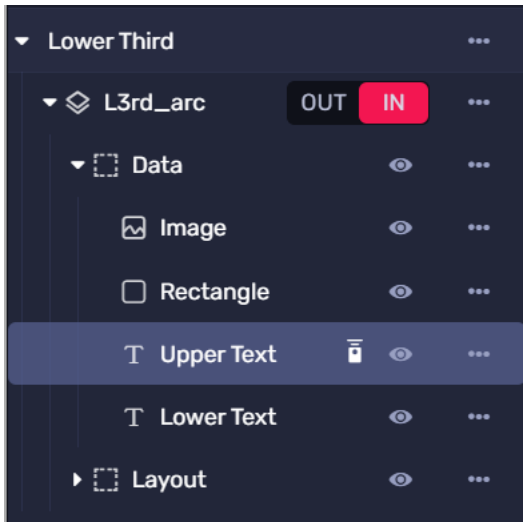
5.21.6 Flowics

Preparing Flowics Graphics for Use in Viz Arc



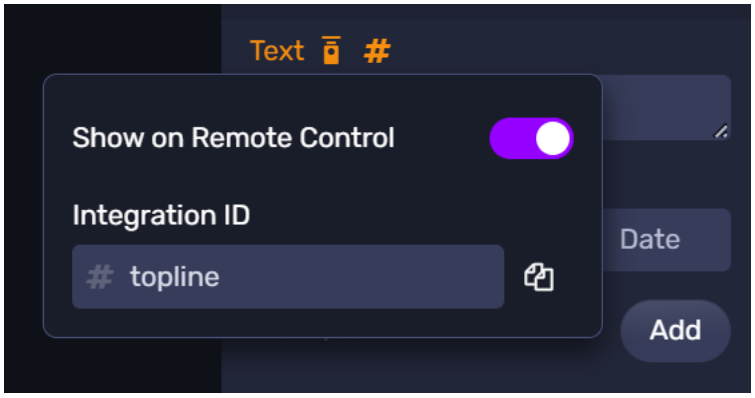
As a first step, the Flowics graphics package needs to be prepared for being controlled by Viz Arc. The steps are the same as they are for controlling the graphics Flowics Remote Control or Rundown Control. The fields that need to be exposed, in this sample the two text lines of the lower third graphics and the image:

Select the graphics element to be exposed.



Enable the **Show on Remote Control** checkbox and add a **Integration ID**.

Copyright © 2025 Vizrt

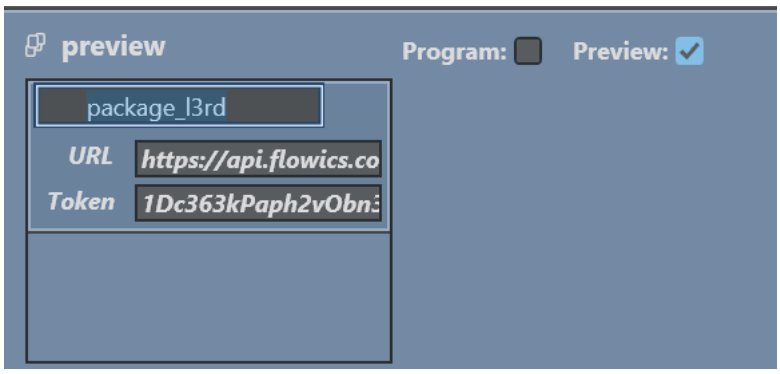


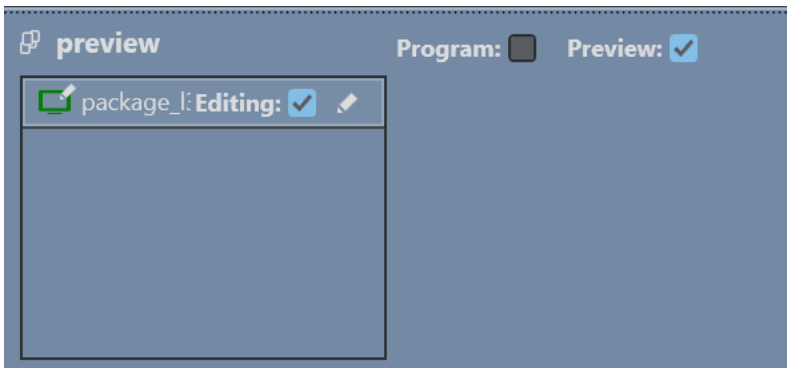
Repeat those steps for all elements that need to be controlled. When done, make sure the modifications are **Published** 

.

Creating a New Flowics Template

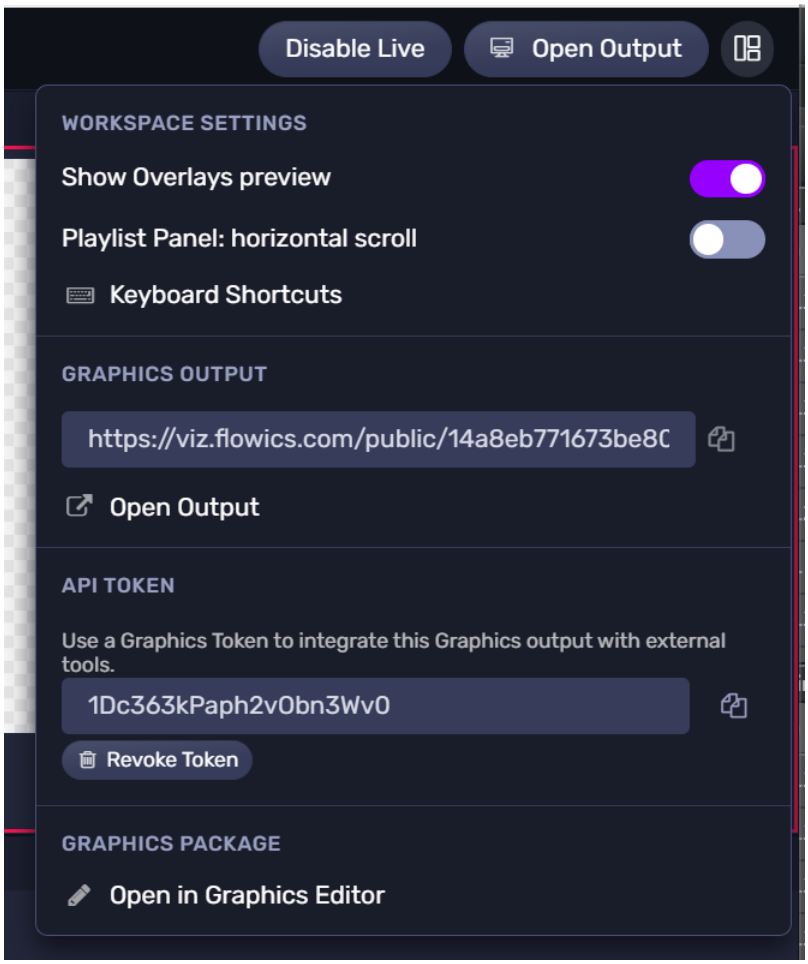
Make sure the editing engine in your profile is configured to use the graphics token of your graphics package.





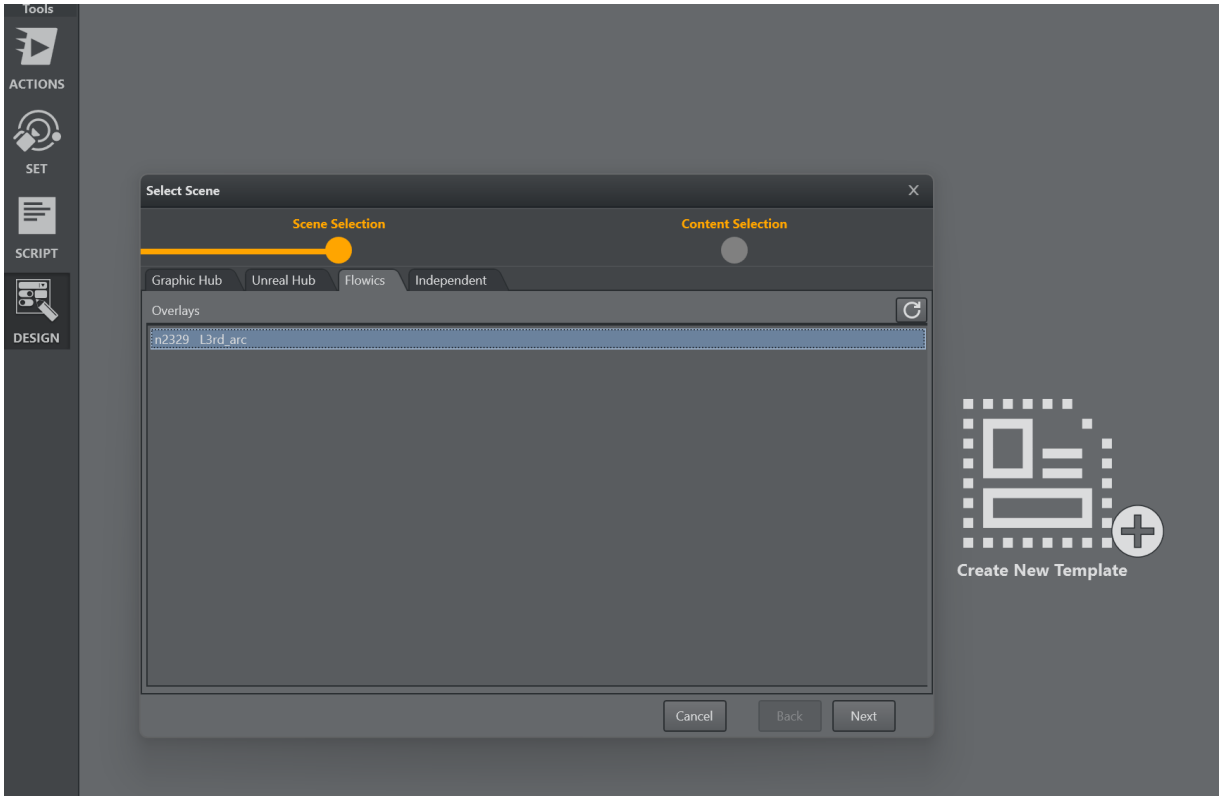
It must match the API Token specified in Flowics:

Copyright © 2025 Vizrt



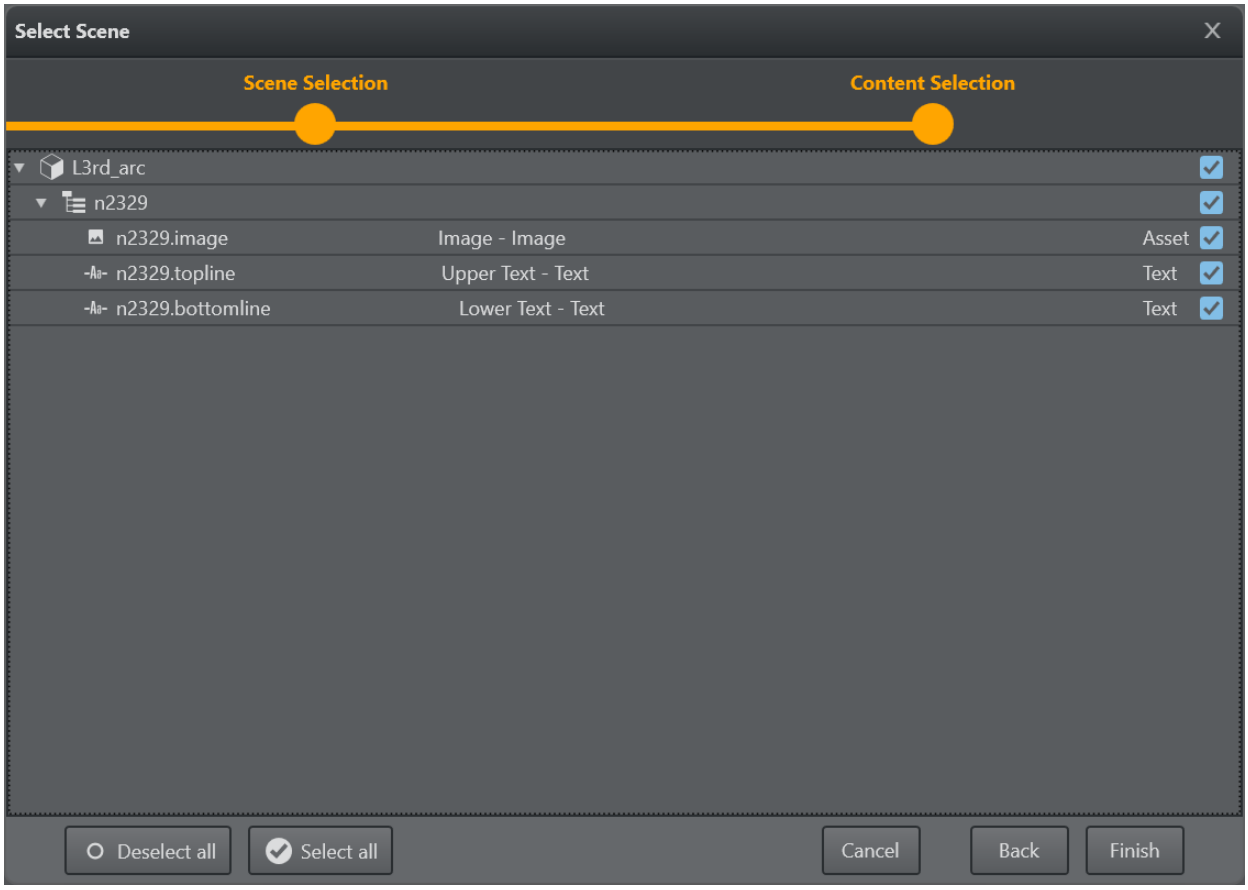
Next, go to the Viz Arc **Design** tab and create a new template. Select the **Flowics** tab an select the overlay to be used.

Copyright © 2025 Vizrt

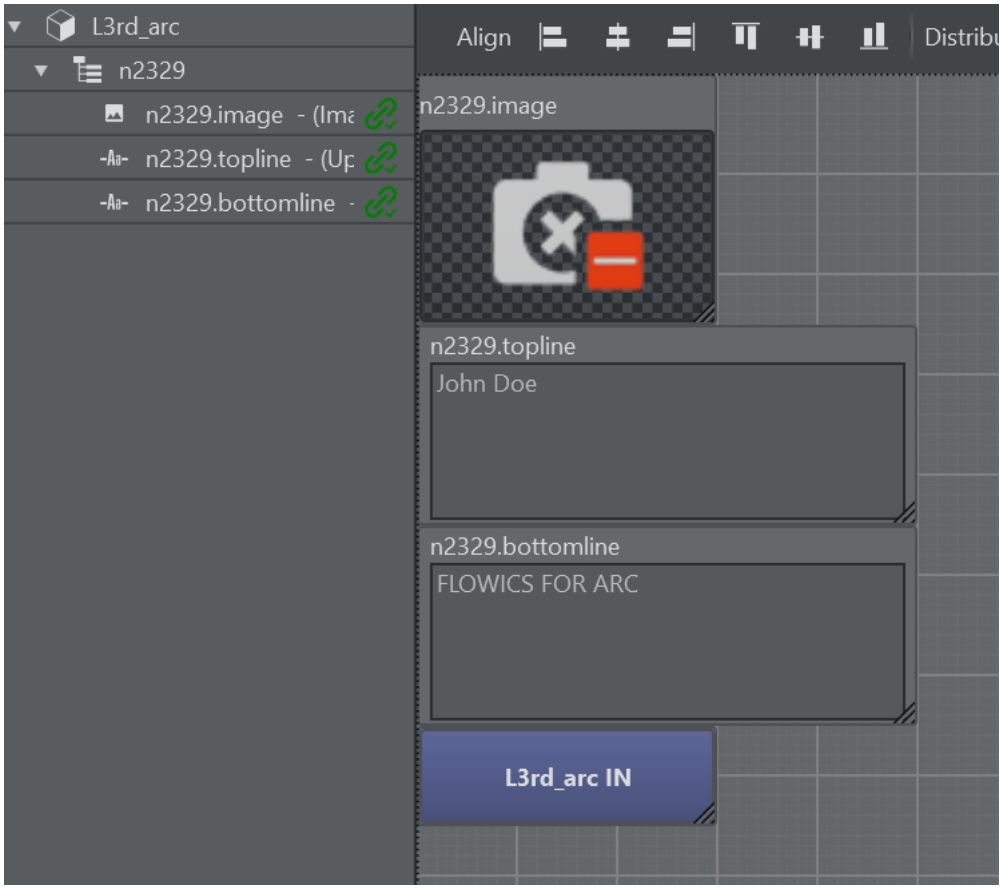


Click next and select the Integration ID's of the overlay to be generated into an initial template UI.

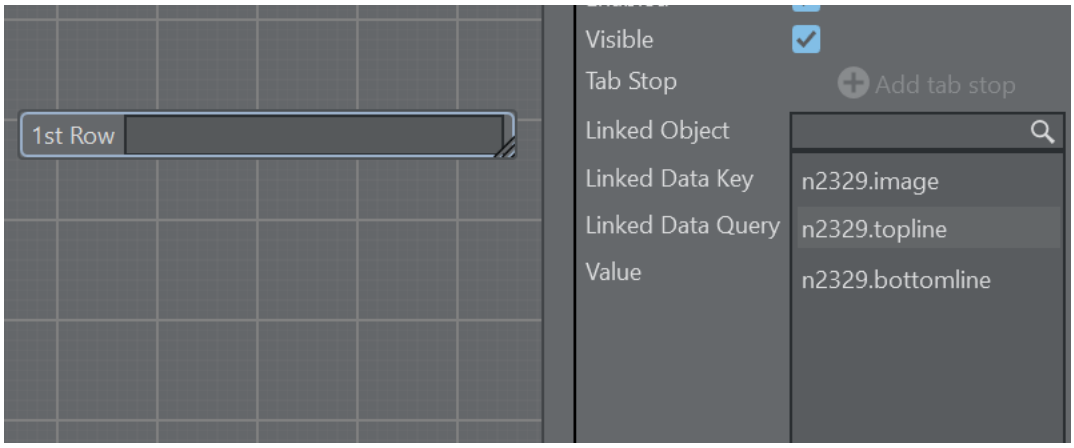
Copyright © 2025 Vizrt



Copyright © 2025 Vizrt

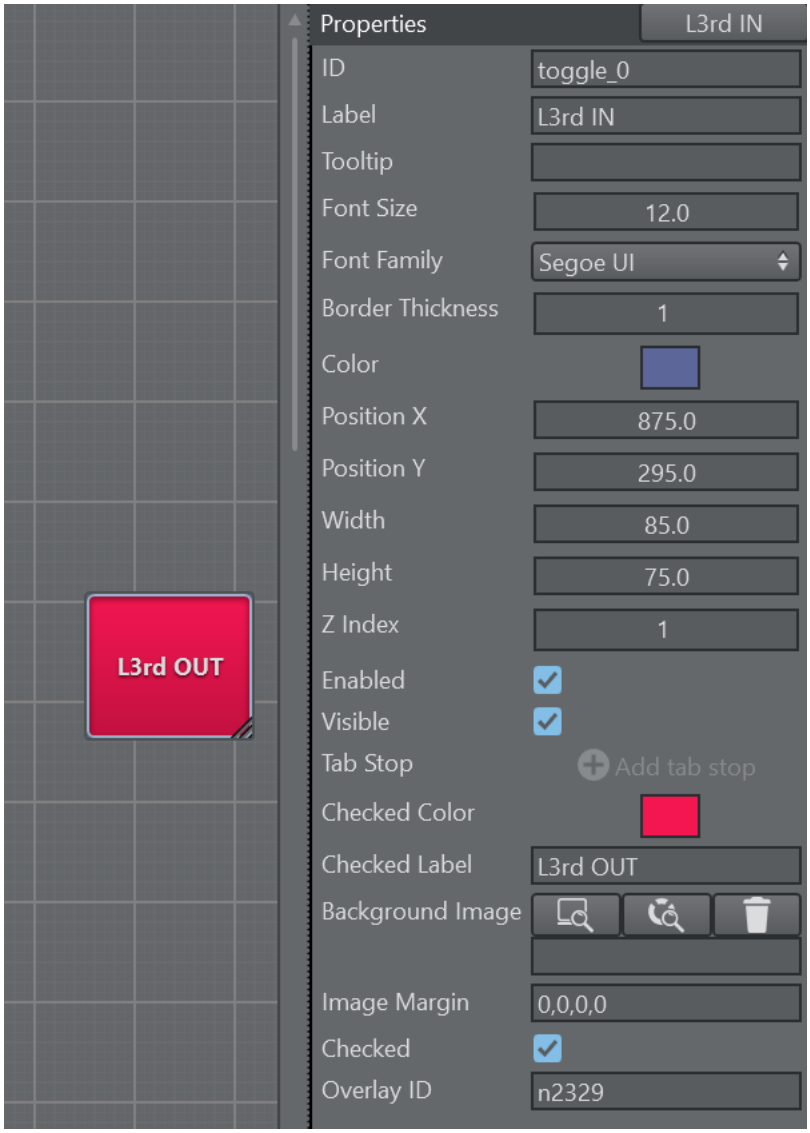


You can also create new UI parameters and link the Integration ID's afterwards. The id's are available the drop down menu as **Linked Object** property.

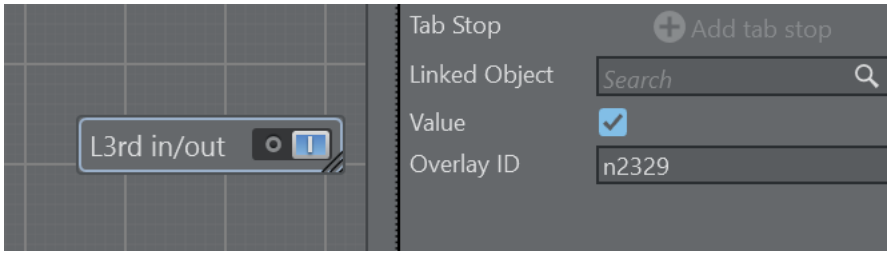


The toggle button responsible to animate the overlay in and out can be customized, the **Overlay ID** can be specified as well as the on/off colors and labels.

Copyright © 2025 Vizrt

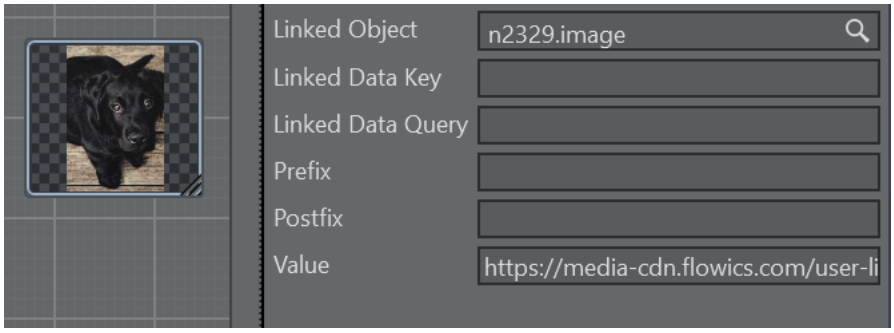


The overlay can be also controlled by a bool parameter, just enter the Flowics overlay id int the Overlay ID property.

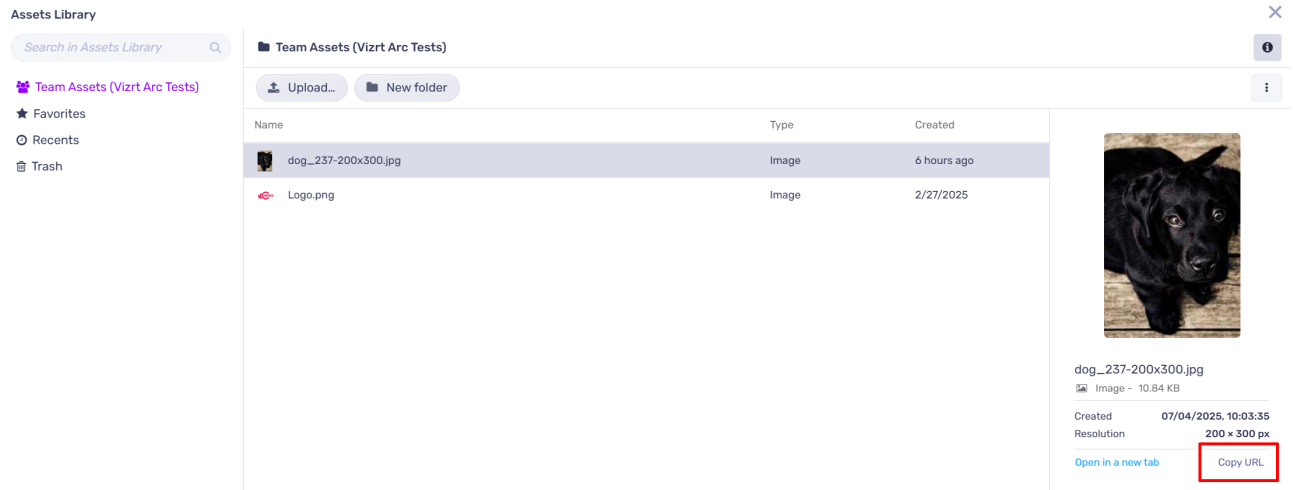


The image Integration ID's can be only assigned manually or through scripting.

Copyright © 2025 Vizrt

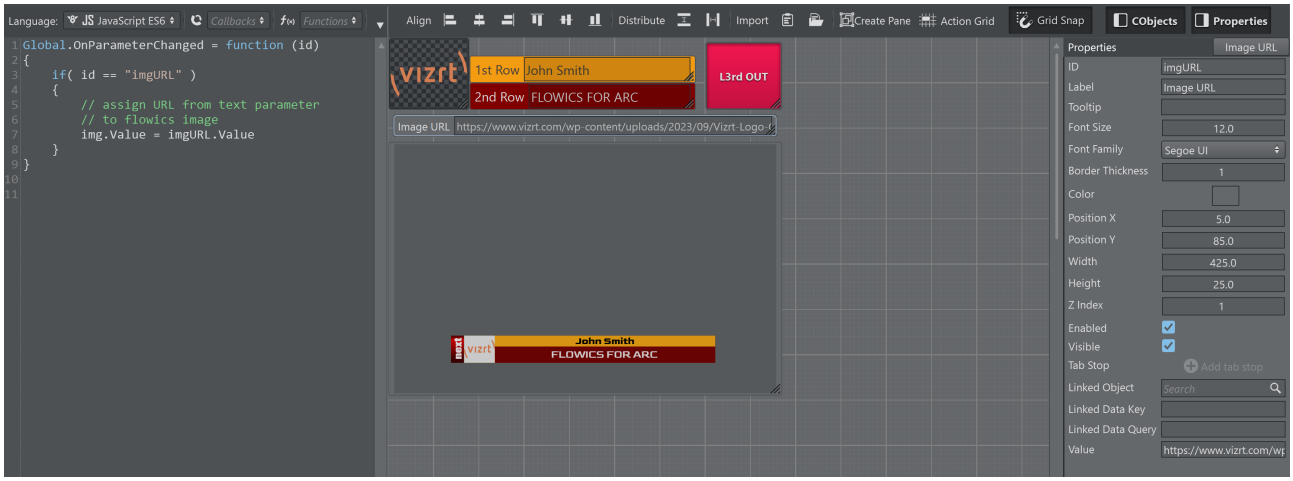


The asset URL can be copied from the Flowics asset library using the **Copy URL** button of the asset

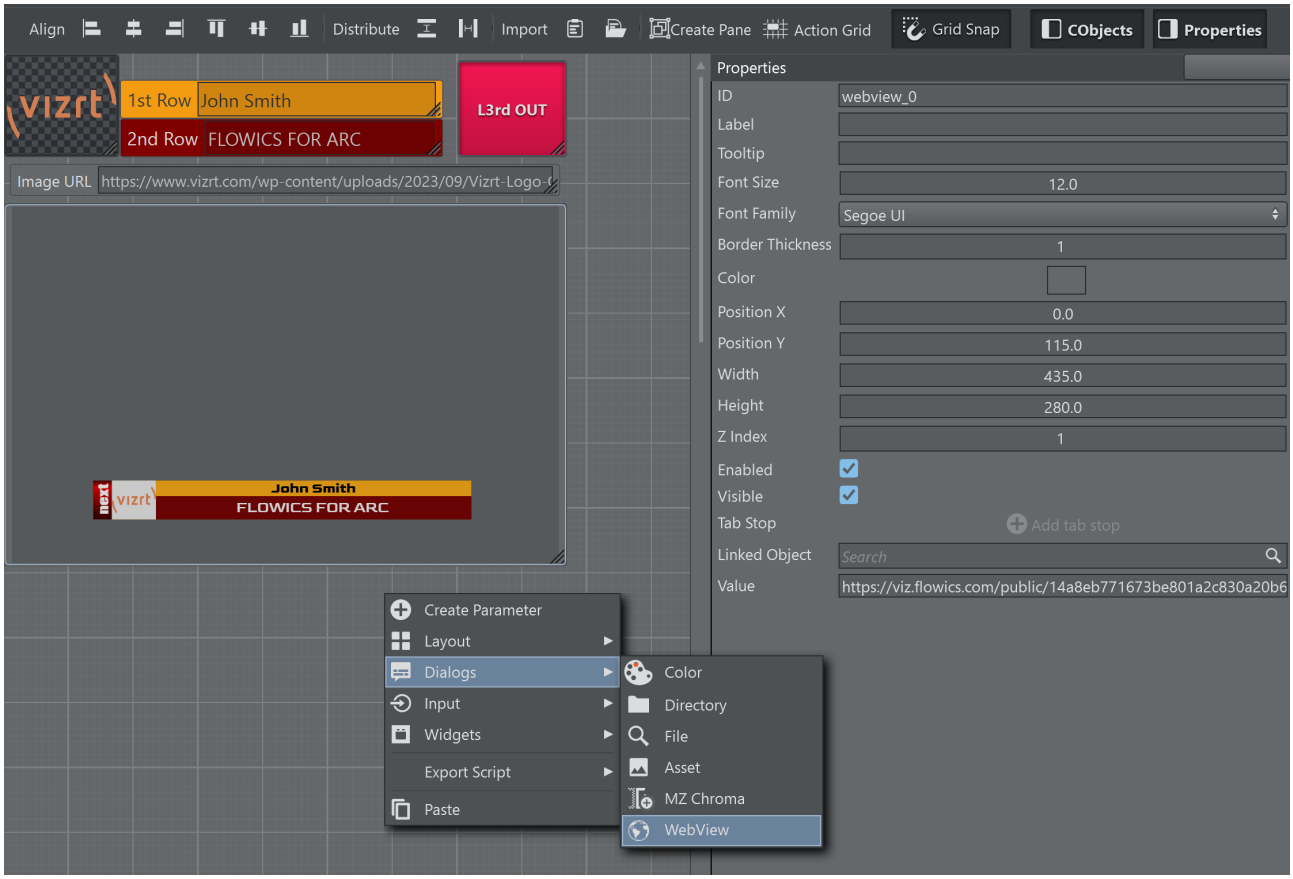


> The Flowics Asset library cannot be browsed from Viz Arc at the moment. Please also note that assets from the Graphic Hub or the local file system will not work. The assets need to come from the library or any other public http endpoint.

A simple script demonstrating how an image URL can be assigned to an asset parameter programmatically.



A **WebView** parameter can be added with the respective output of the Flowics output.



Execution and Update Logic

When a Flowics template gets executed through the **Execute** button or through shortcuts, the Integration ID's mapped in the template UI and the toggles mapped in the UI are updated on the Flowics output.

When the template gets updated through the **Update** button or it's respective shortcut, only the Integration ID's are updated on the Flowics output.

Data can be updated selectively using the **UdpateTemplate** scrpting method. When the function is called without parameters, all Integration ID's mapped in the template UI are updated on the output. When a space separated list of Integration ID's is used, only those ID's are updated on the output.

Extending the sample above, the output is updated as soon as the text is validated on the template UI (though lost focus or on enter pressed in the text parameter).

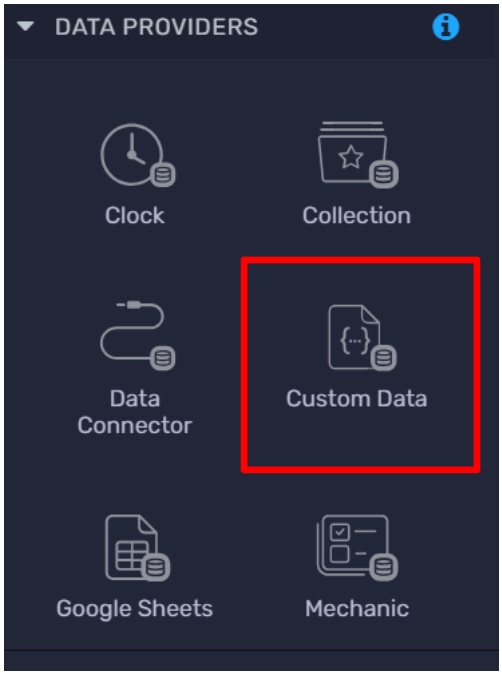
```
Global.OnParameterChanged = function (id)
{
    if( id == "imgURL" )
    {
        // assign URL from text parameter
        // to flowics image
        img.Value = imgURL.Value
    }
    else if( id == "text_2nd" || id == "text_1st")
    {
```

Copyright © 2025 Vizrt

```
        // text_2nd and text_1st are the internal Viz Arc script id's of the text
parameter's
        // linked to the Floics Integrtion ID's n2329.bottomline and n2329.topline
        // only update the 1st and 2nd row text
        UpdateTemplate("n2329.bottomline n2329.topline")
    }
}
```

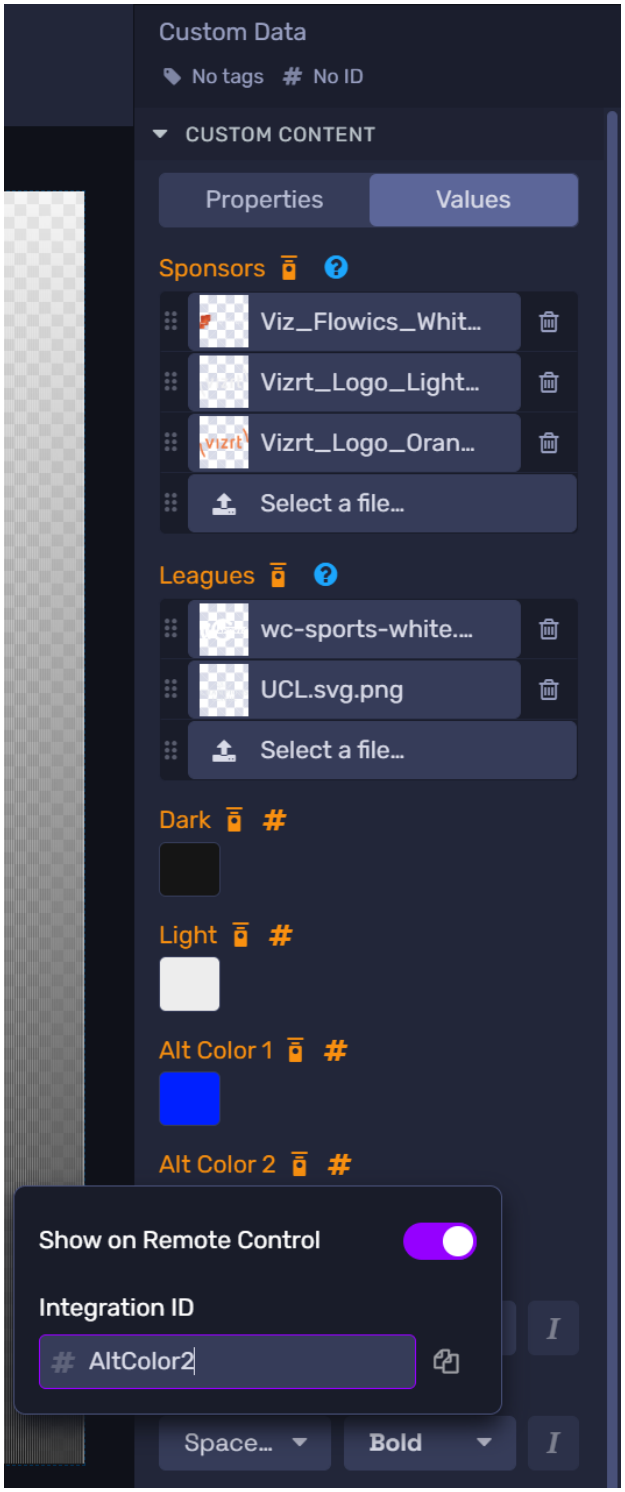
Global Data Provider - Custom Data

The Global Data Provider in Flowics is a powerful tool to control graphic settings throughout the GFX package. A setting (e.g. a text string or a color) can be linked multiple times throughout one or multiple overlays.



In particular the **Custom Data** provider can be used to expose controls to Viz Arc templates.

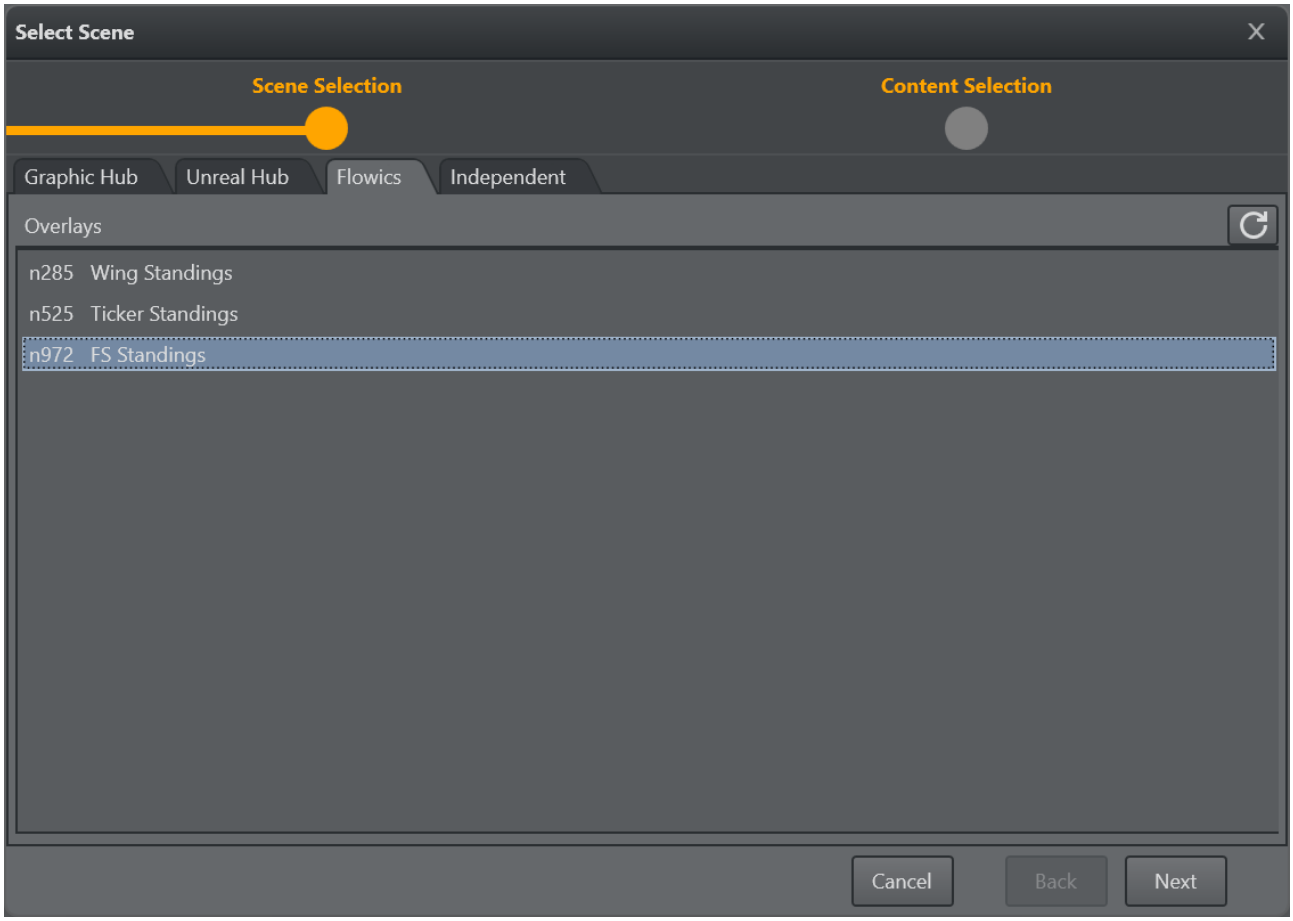
Copyright © 2025 Vizrt



The respective control needs to have an Integration ID assigned to be exposed to Viz Arc. Currently only Text, Image and Color properties are supported.

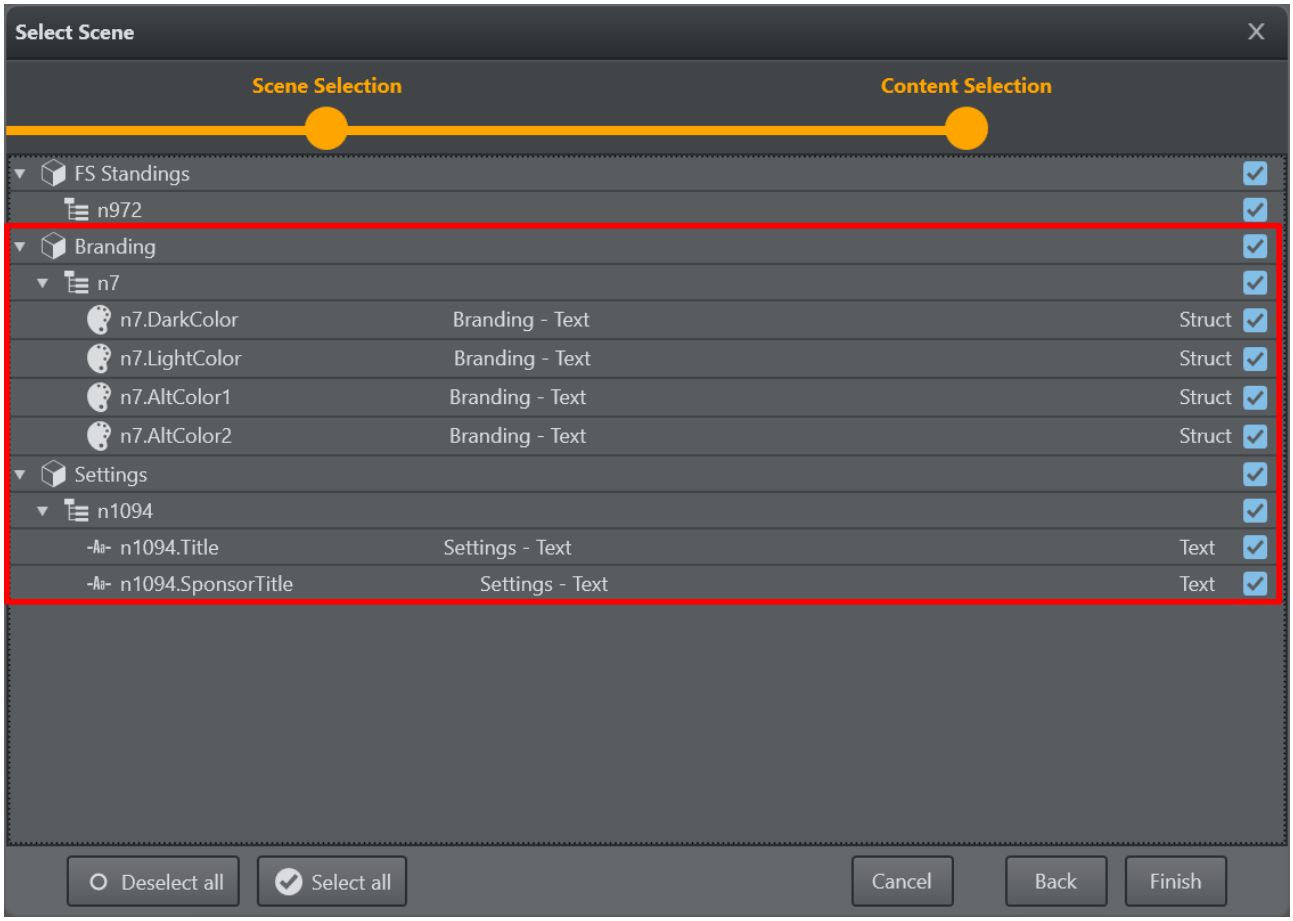
Create a new template and select the Overlay(s)

Copyright © 2025 Vizrt



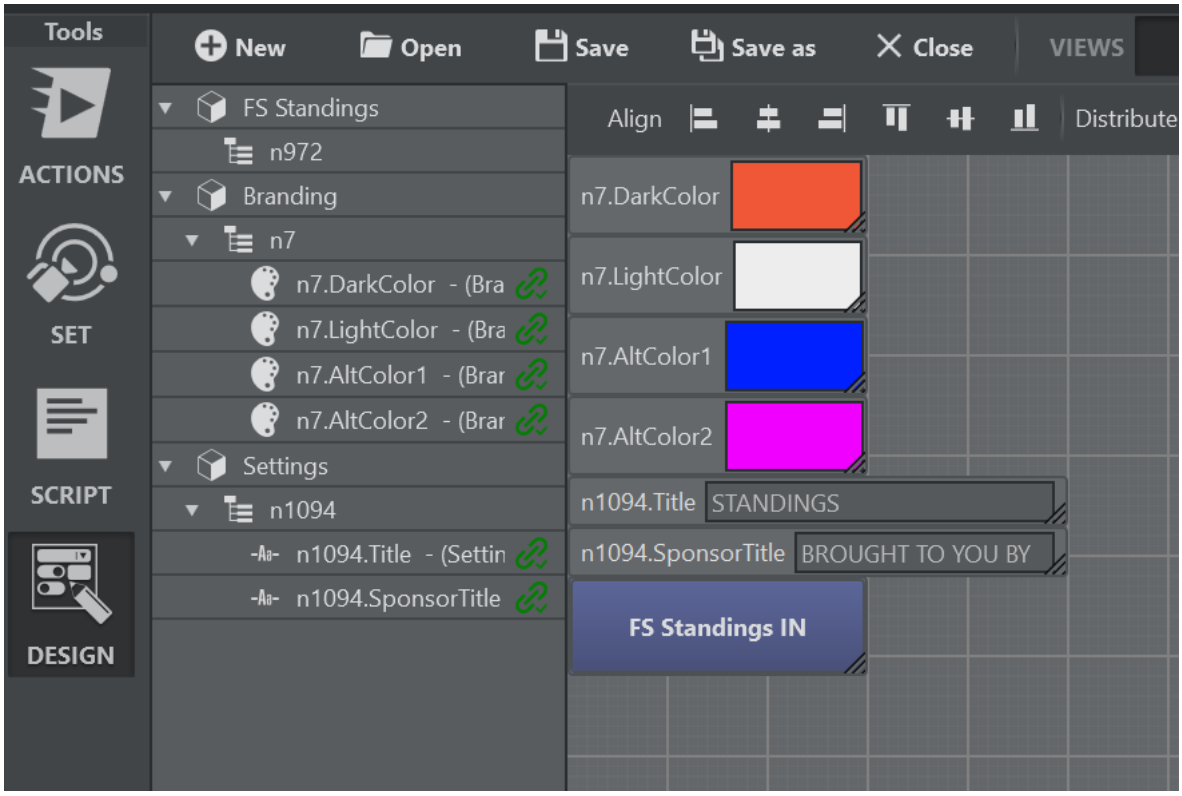
In the next step of the template wizard the Global Data Providers show up where the single elements can be selected/deselected for automatic UI generation.

Copyright © 2025 Vizrt



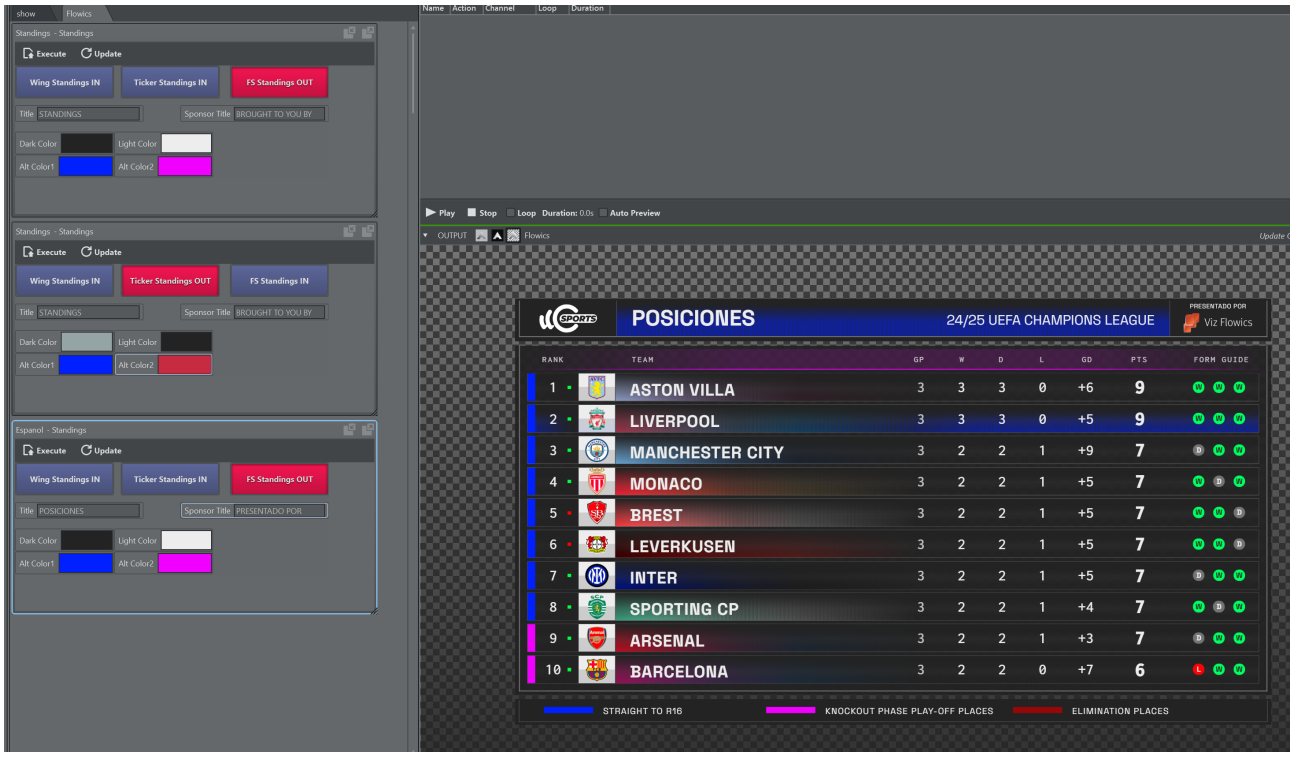
The UI generated might then look like this:

Copyright © 2025 Vizrt



The logic of Execute and Update

Once the template is finished it is ready to be used.

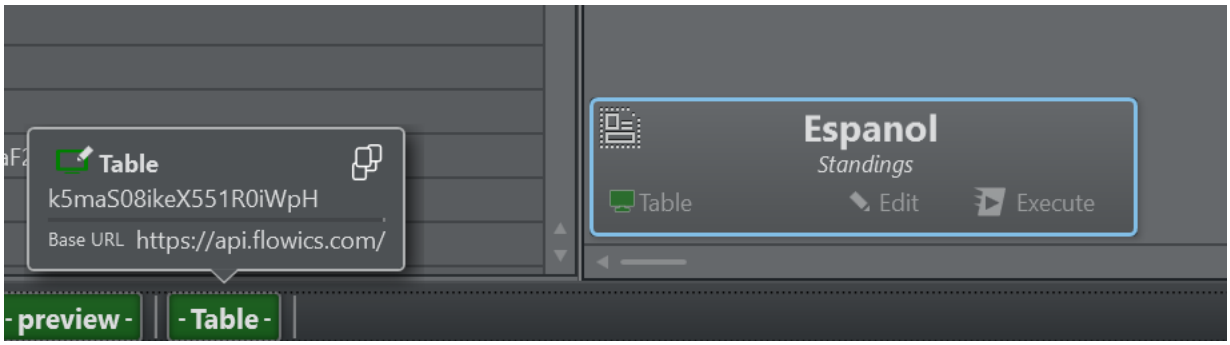


Copyright © 2025 Vizrt

When a template is **Executed** all the linked data in the template is updated on the Flowics graphics and all the toggle buttons present in the template are going to be matched in the graphics. Overlays are animated in or out according to their status in the template UI.

When a template is just **Updated**, only the linked data items are updated on the Flowics graphics.

Make sure the templates are linked to the correct output channel/engine matching the correct Flowics API token.



Executing Custom Commands

It is possible to send custom commands to Flowics engines. They need to be wrapped in a JSON structure, as shown below, and sent to the engine as a stringified JSON.

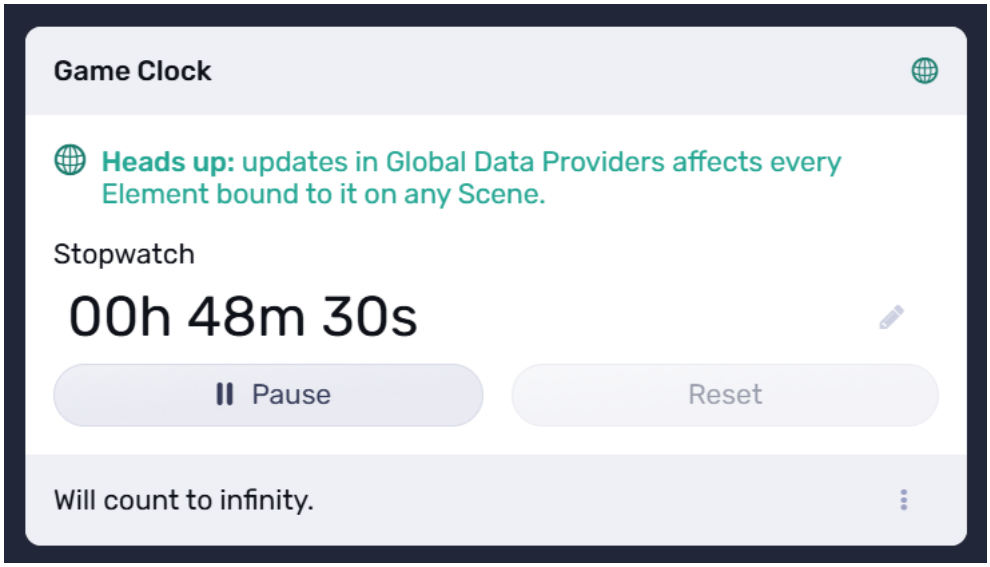
```
{
   "command" : string,
   "method" : string,
   "postfix" : string,
   "content" : string,
   "doArrayWrap" : bool
}
```

- **command:** Needs to be “custom”.
- **method:** Can be PATCH, GET, PUT, POST, DELETE.
- **postfix:** The postfix to be added to the URL.
- **content:** The JSON content to be sent as a payload to Flowics.
- **doWrap:** Use true to wrap the content into JSON array brackets ‘[' and ']’.

Info: The Flowics API documentation can be found here.

The clock as it appears in the Flowics remote control:

Copyright © 2025 Vizrt



The following example shows a convenience function for a Viz Arc Javascript, that sets and starts a soccer clock (for example, it can be triggered when a match starts at 00:00 or the second half at 45:00).

```
function SetClockAndPlay(clockVal)
{
    var flowicsCommand = new Object()
    flowicsCommand.command = 'custom'
    flowicsCommand.method = 'PATCH'
    flowicsCommand.postfix = '/control/global-data-providers'
    flowicsCommand.content = "{\"id\":\"n1818\",\"controls\":{\"stopwatch\":
{\"value\":{\"startFrom\":\"" + clockVal +"\",\"timeReference\":\"" + Date.now() +
"\",\"current\":\"" + clockVal + "\",\"state\":\"play\"}}}}"
    flowicsCommand.doArrayWrap = true

    GetSelectedChannel().SendSingleCmd(JSON.stringify(flowicsCommand))

    Console.WriteLine("SetClockAndPlay " + flowicsCommand.content)
}
```

The function above can be invoked as:

```
SetClockPlay("00:45:00")
```

The *content* as a JSON:

```
{
   "id":"n1818",
   "controls":{
      "stopwatch":{
         "value":{
            "startFrom":"00:45:00",
```

Copyright © 2025 Vizrt

```
            "timeReference":"1753173655377",
            "current":"00:45:00",
            "state":"play"
         }
      }
   }
}
```

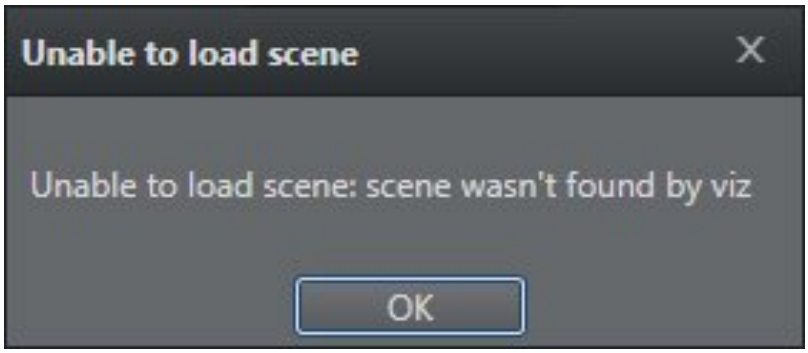
6 Troubleshooting

This page covers the following topics:

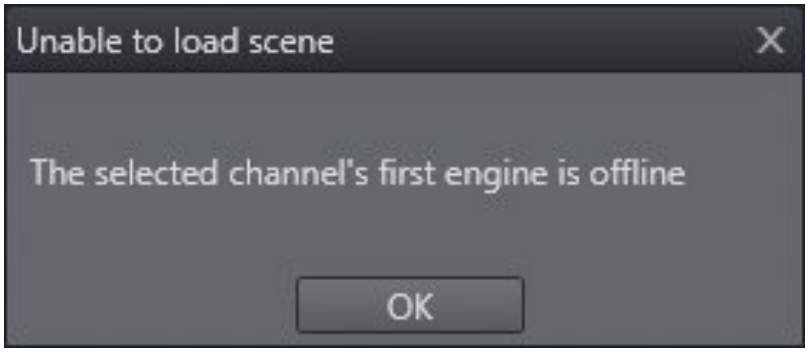
- Vizrt Issues
  - Unable to Load Scene
  - Graphic Hub Not Connected
  - Preview Not Connected
  - Output Initialize Failed
  - Cannot Connect/Send to Viz Engine
  - Cannot Send Shared Memory Commands
- Unreal Engine
  - Unreal Engine not Available
  - Cannot Connect/Send to Unreal Engine
  - Unable to Load Unreal Level
- NDI
- Support

6.1 Vizrt Issues

6.1.1 Unable to Load Scene

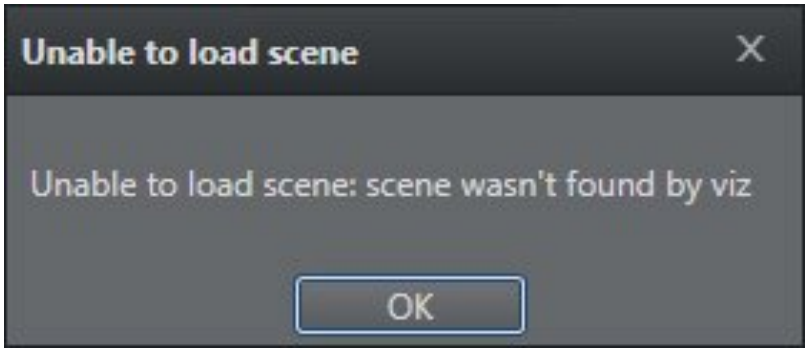


Make sure that there is a channel associated to a scene, and that you are connected to a Vizrt profile.



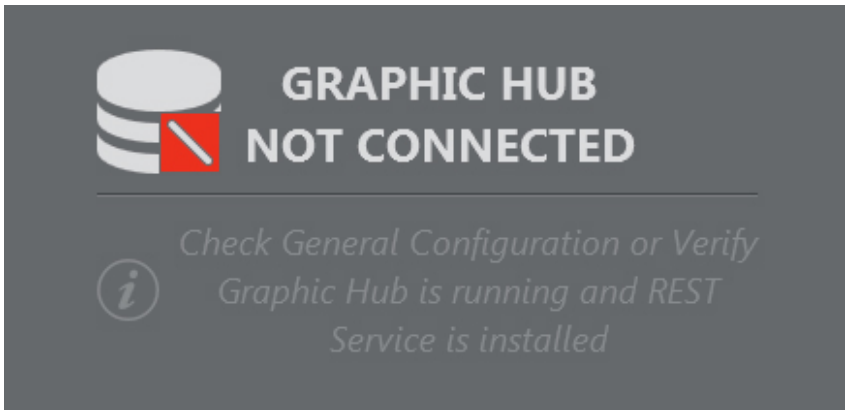
Check if Viz Engine is up and running.

Copyright © 2025 Vizrt



Viz Arc cannot find the selected scene. Check that you are connected to the correct Graphic Hub.

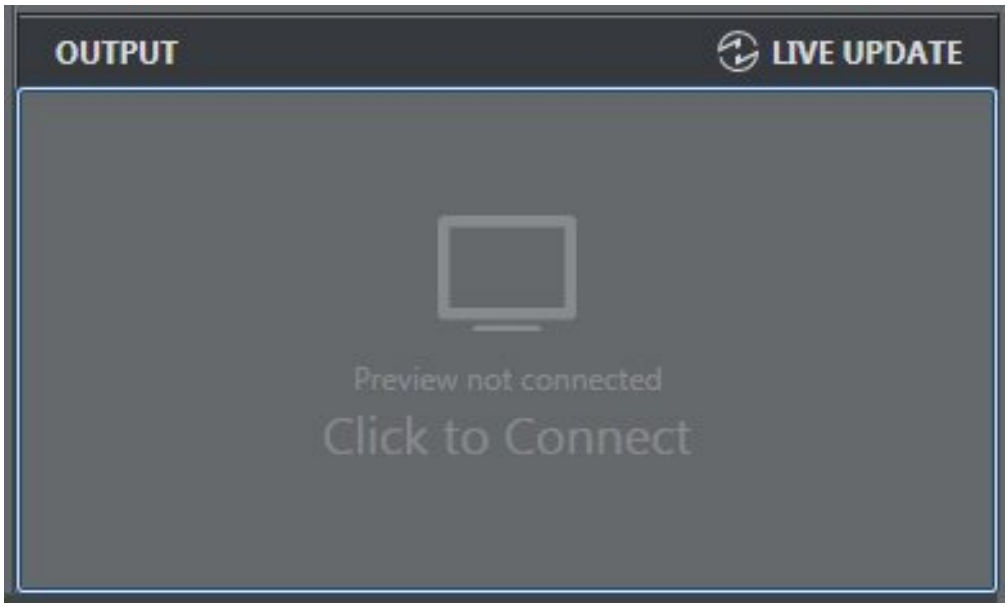
6.1.2 Graphic Hub Not Connected



- Check if Graphic Hub is up and running.
- Check in the **Config** window that the Graphic Hub REST settings are correct (Host/IP and Port, Username and password).
- Check that the machine on which Viz Arc is installed is connected to the machine containing Graphic Hub.
- Follow the procedure in General Configuration to correctly configure Graphic Hub REST in Viz Arc.

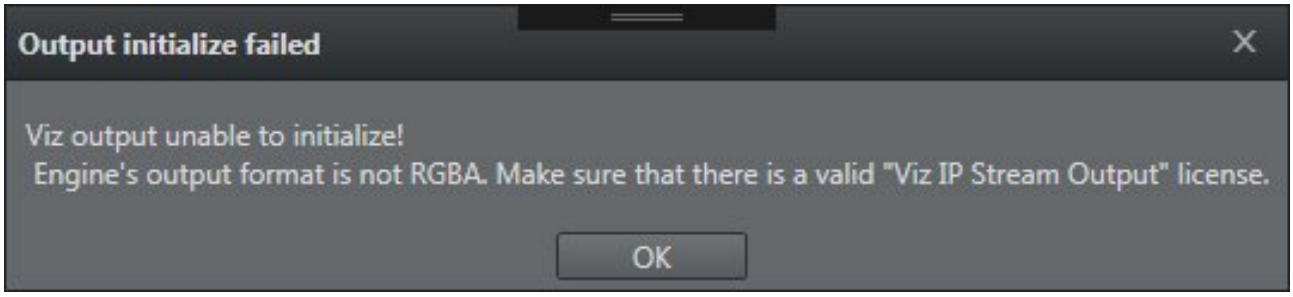
Copyright © 2025 Vizrt

6.1.3 Preview Not Connected



- Check if Viz Engine is up and running.
- Check in the Config window that the Viz GH and MSE settings are correct (Host/IP and Port).
- Check that the machine on which Viz Arc is installed is connected to a Viz Engine.
- See the section Configuring Viz Engine.

6.1.4 Output Initialize Failed



- Check that Viz Engine is configured in RGBA output format.
- Ensure that there is a valid Viz IP Stream Output license.
- Check that you are using Viz Engine 3.9 or higher.
- See the section Configuring Viz Engine.

Copyright © 2025 Vizrt

6.1.5 Cannot Connect/Send to Viz Engine



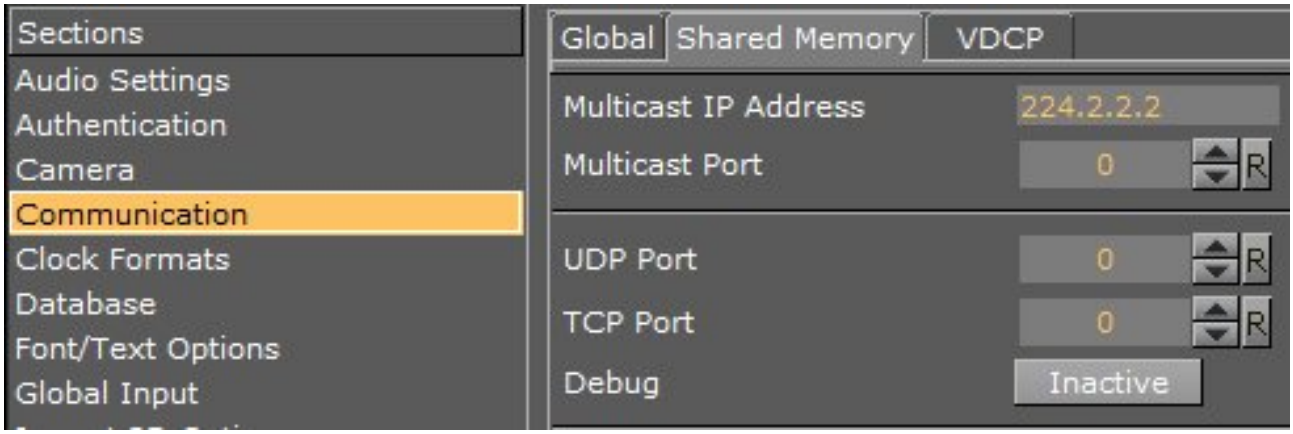
Check that Viz Engine is up and running and set in On Air mode.



Check that Viz Engine is set in On Air mode.

6.1.6 Cannot Send Shared Memory Commands

If Viz Engine can't receive SHM commands, it's possible that the UDP and TPC ports are not configured in Viz Config. In Viz Engine, go to **Config > Communication > Shared Memory**.

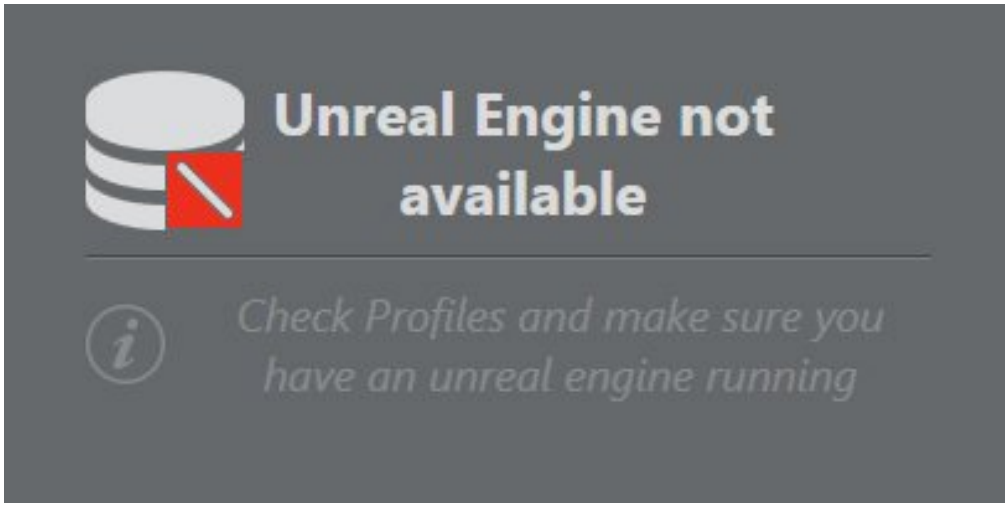


For more information, see Profiles.

Copyright © 2025 Vizrt

6.2 Unreal Engine

6.2.1 Unreal Engine not Available



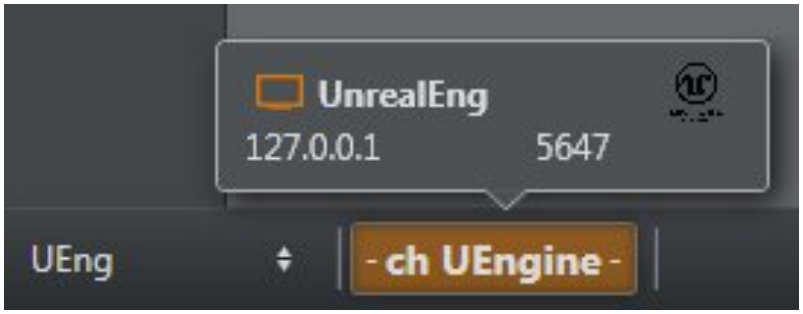
- Check that Unreal Engine is up and in Play Mode.
- Check in the Config window that the Unreal Engine settings are correct (Host/IP and Port).
- Check that the Viz Arc machine is connected to the machine containing Unreal Engine.
- Check that you have copied the *arcCom* plug-in folder to *\Epic Games\UE_4.20\Engine\Plugins*.
- See Profiles for more information on configuring Unreal Engine in Viz Arc.

6.2.2 Cannot Connect/Send to Unreal Engine



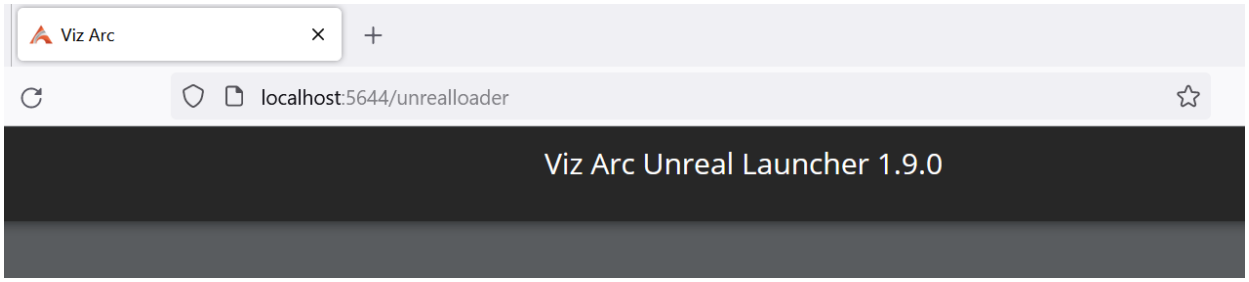
Check that Unreal Engine is up and running and set in Play mode.

Copyright © 2025 Vizrt

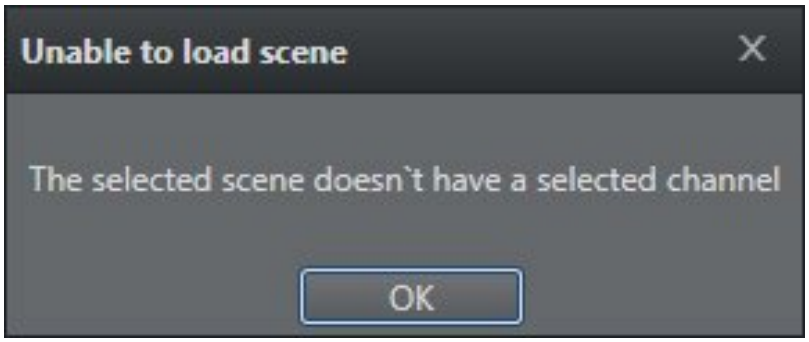


Check that Unreal Engine is set in Play mode.

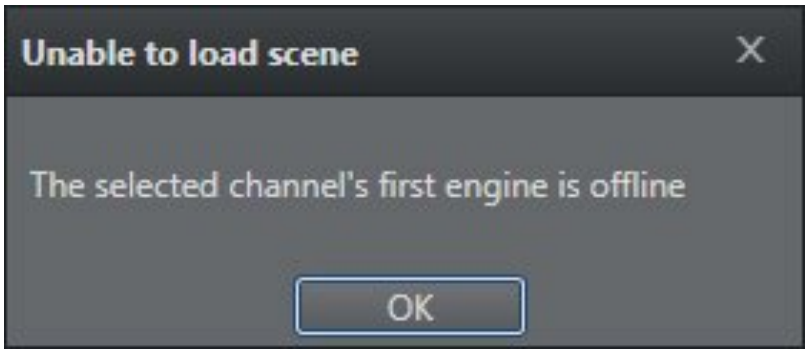
Check if the Unreal Launcher service is running and responding by pointing a web browser to http://hostname:5644 where 5644 is the launcher port.



6.2.3 Unable to Load Unreal Level

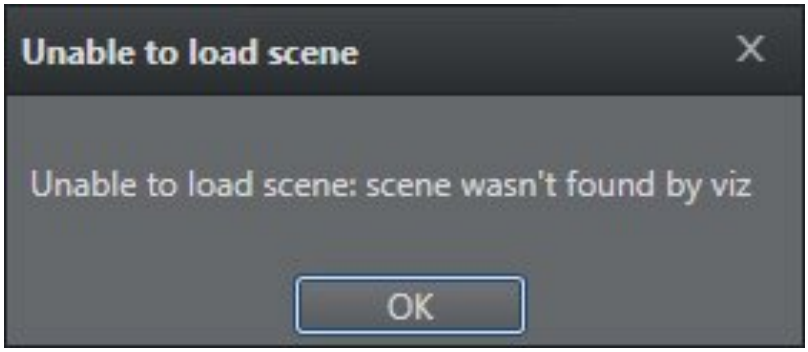


Ensure that there is a channel associated to a Level, and that you are connected to an Unreal Engine profile.



Check that Unreal Engine is running and in Play mode.

Copyright © 2025 Vizrt



Viz Arc cannot find the selected Level. Check that you are loading a Level that exists in the selected project.

6.3 NDI

Viz Arc gives priority to the installed NDI runtime library which is stored in the environment variable **NDI_RUNTIME_DIR_V5**. If this environment variable is present, all NDI related dll's are loaded from this folder.

If **NDI_RUNTIME_DIR_V5** is not present, Viz Arc will check for **NDI_RUNTIME_DIR_V4** and load the dll's from there in case the folder exists. If none of the above environment variables are set, Viz Arc uses the shipped NDI dll's which can be found in the Viz Arc installation folder (for example *C:\Program Files\Vizrt\Viz Arc 1.9\NDI*).

In some cases the installed NDI libraries might cause incompatibilities with Viz Arc resulting in NDI video streams not visualizing although the streams are seen by Viz Arc. In those cases you can force Viz Arc to use the shipped dll's by copying all the content of *C:\Program Files\Vizrt\Viz Arc 1.9\NDI* into the main program folder *C:\Program Files\Vizrt\Viz Arc 1.9.* Administration privileges are required and a restart of the application is needed to apply the changes.

6.4 Support

Support is available at the Vizrt Support Portal.

Copyright © 2025 Vizrt