Release Notes Vertex 2025 R2

NEW FEATURES

Destinations ▸
Compositions ▸

Live Input Preview in Remote Render Editor ▸

Stream Deck Editor ▸

Notch 1.0 support

Project Bundle ▸

Parameter Clips ▸

Control View Web Forms Mode ▸
Control View: receive IP address of current Web Client ▸
Control View Button Icons ▸
Control View Button Gradients ▸
Control View Controls Margin ▸
Control View: Label Trimming and Wrapping ▸

MIDI TimeCode Out ▸

Surface Modes (property filtering) ▸

Audio Channel Map Matrix ▸

Projector Study Data Import ▸

Auto-Generate Surfaces from Geometry ▸

Logitech MX Creative Console Device ▸
Analog Way Live Premiere Device

Shotbox "All Solo" mode ▸

Cue "Fade To" Modes ▸
Cue "Run Script on Play" ▸

Advanced Multi-System GPU Sync ▸

Various scripts added for content management ▸
ClipContainer Enter/Leave Script added ▸

"Frame Time" in System Fullscreen Status ▸

FIXES

Notch Blocks:
- handling optimizations
- Aux processing optimizations
- multiple instances of same block support added

Resync Video Drift optimizations
Displayrate Mode optimizations

Tiff Image Sequence fixes and optimization
Tiff Macintosh byte order support added

Control View:
- DropDown Control web index offset fixed
- Calender Control load items color fixed
- controls copy/past fixed
- Image Control PSD and PPT Sub-Contents fixed
- new Webpolling update interval property
- Prompt Window size adjusted for Web View

Stype Device init fixed
Schauf Game Clock Device Reset Packet Counter on Reconnect

Mjpeg streaming fixes and optimizations
Mjpeg streaming settings moved from Video/Streaming Control to LiveInput Content

Playback Editor zoom fixed
Playback Logging changed to start logging playback > 0.5s only

Clipcontainer "Use Basic Shader" property replaced with "Shader Performance" dropdown property
RenderInView only fixed
FadeToTime fix for Transform Keyframes
Texture smoothing optimizations – improves image quality on scaling

Canvas and Visualizer light source improvements

SMPTE timecode fixes (check SMPTE I/O firmware update!)

Remote Installer fixed runtime >3 days

several fixes and optimizations on
- Playback/ Cue behavior
- Shared Textures
- Playlists
- Shotboxes
- DataTables
- NestedSequences
- MarkerCalibration
- Inspector
- SyncClock
- Content Transfer

Output Stream Multi Canvas Audio speed fixed

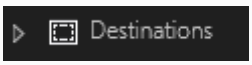
Softedge Normal Mode Luminance default set to 1
Softedge Square Mode Luminance and Gamma swapped plus luminance added for processing

Surface Testpattern rendering memory leak fixed

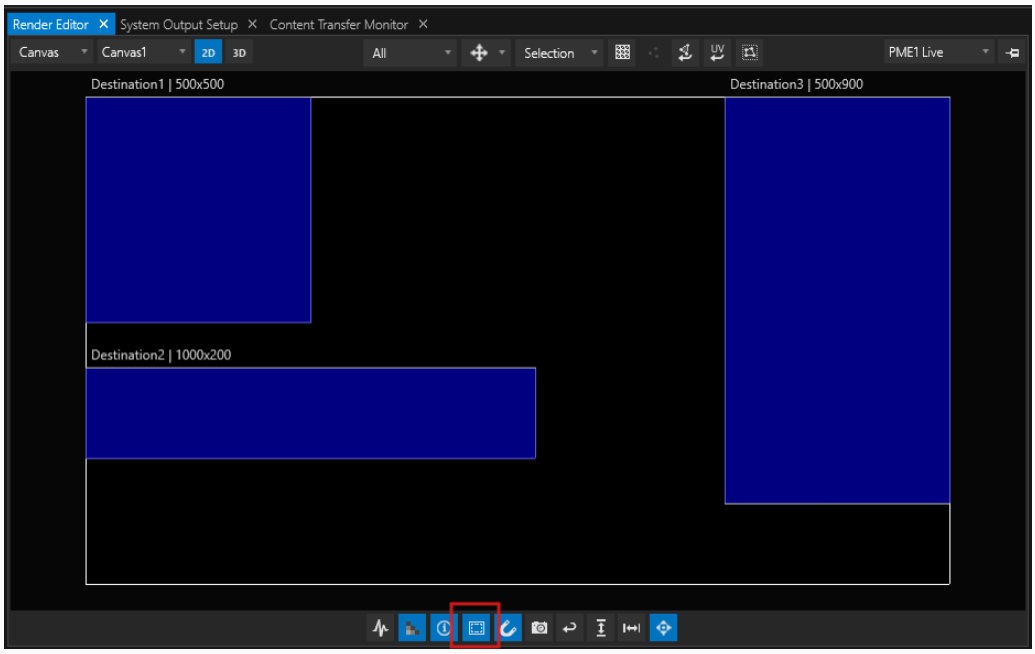
Destinations

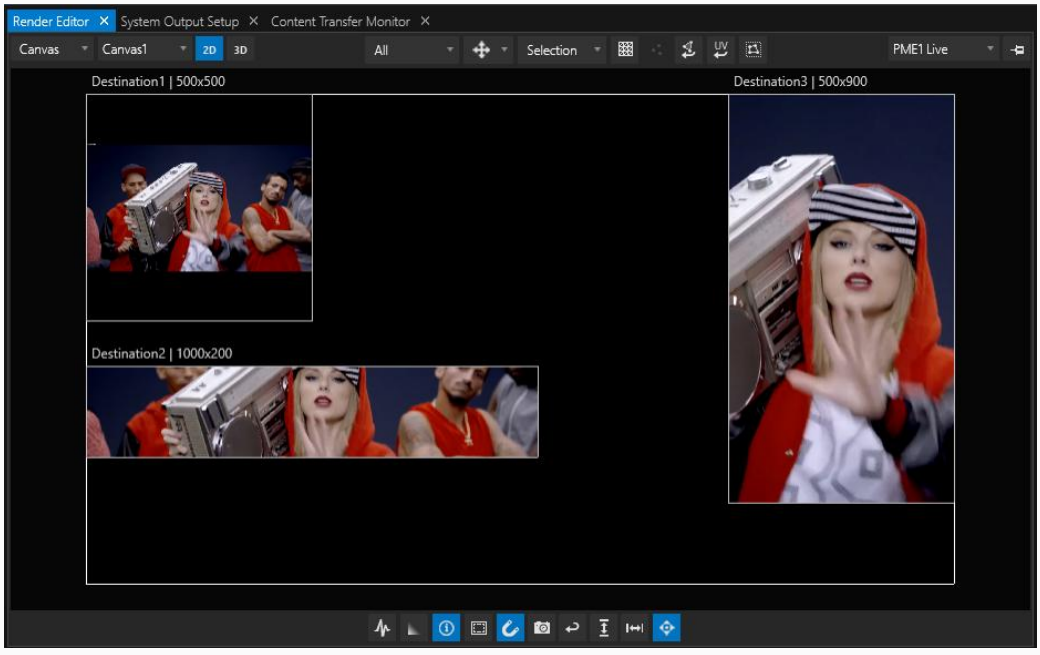
With Destinations, Vertex introduces a simplified workflow for quickly setting up and managing LED screens or simple displays. Unlike the advanced canvas surface concept, which is designed for complex XR studios or projection mapping, Destinations allow for fast configuration of straightforward layouts.

For these cases, Destinations make it possible to define outputs with just a few basic parameters such as name, position, and size (on both Canvas and Output).

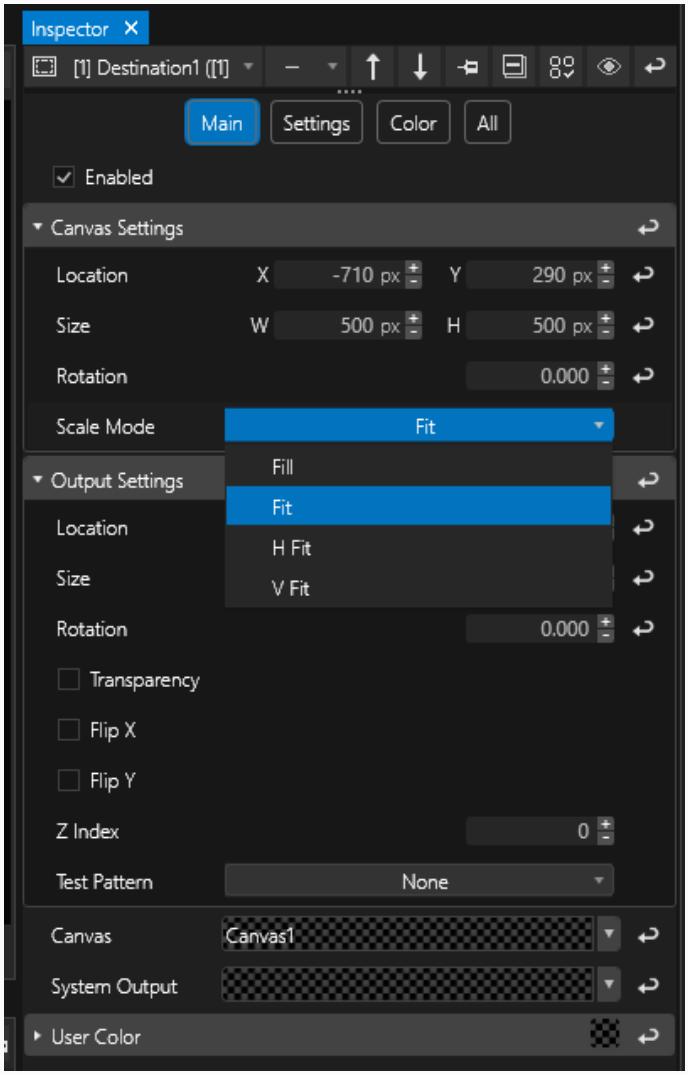


Content can be assigned directly onto Destinations **by drag and drop in the Render Editor**, automatically creating Clips on the current selected Playback. (Activate the Render Editor "Show Destinations" mode (see screenshot) and drag Contents onto the blue rectangles)





Every Destinations has its individual Scale Mode (Fill, Fit, Horizontal Fit or Vertical Fit). A Clip applied to the Destinations will automatically get scaled to the size of the Destination and positioned to the Canvas Location of the Destination. If the Destination is moved or resized, the assigned Clip follows accordingly, which makes adjustments fast and reliable. Destinations can also be positioned, rotated, or flipped freely on system outputs, and a Z-index allows layering in depth.



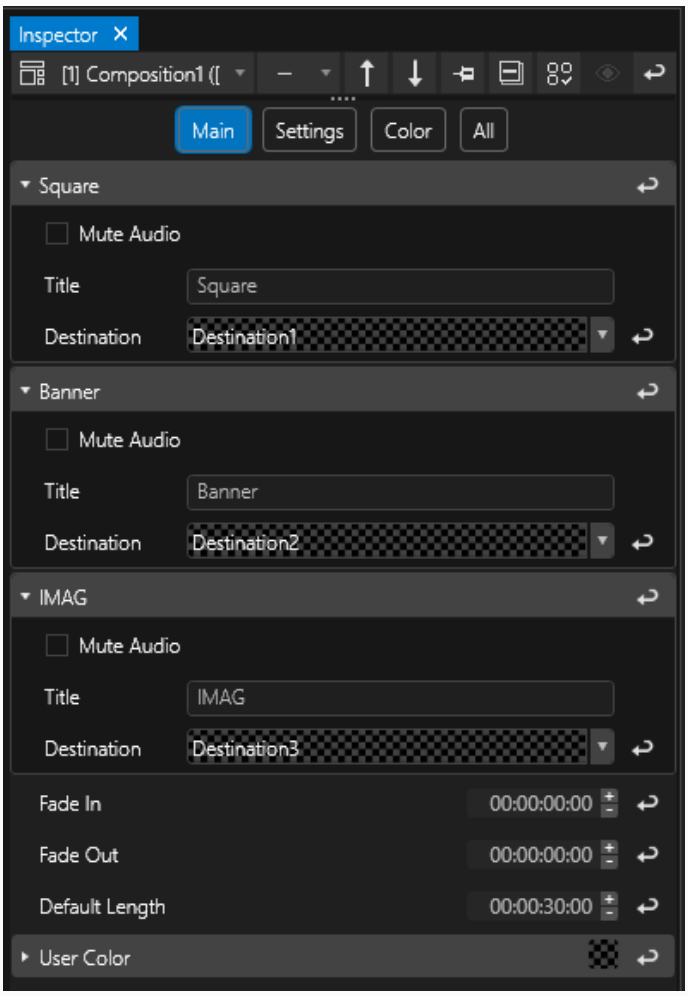
Clips or entire tracks can be assigned directly to Destinations as their Clip Target. This is different from Surfaces, as they cannot serve as a Clip Target. Using a Destination as a Clip target will combine both – routing the Clip to a specific Canvas and making sure the Clips will be rendered on the Output, specified inside the Destination.

Destinations provide a faster, more intuitive alternative when setting up simple screen configurations while still integrating seamlessly with the existing Canvas/Surface concept.

Compositions

Compositions build on Destinations by grouping them into reusable layouts, making it easier to manage multi-screen timelines.

Slot-Based Layouts - A Composition consists of slots, each assigned to a Destination. Slots define where Content is placed, but do not contain Content themselves.



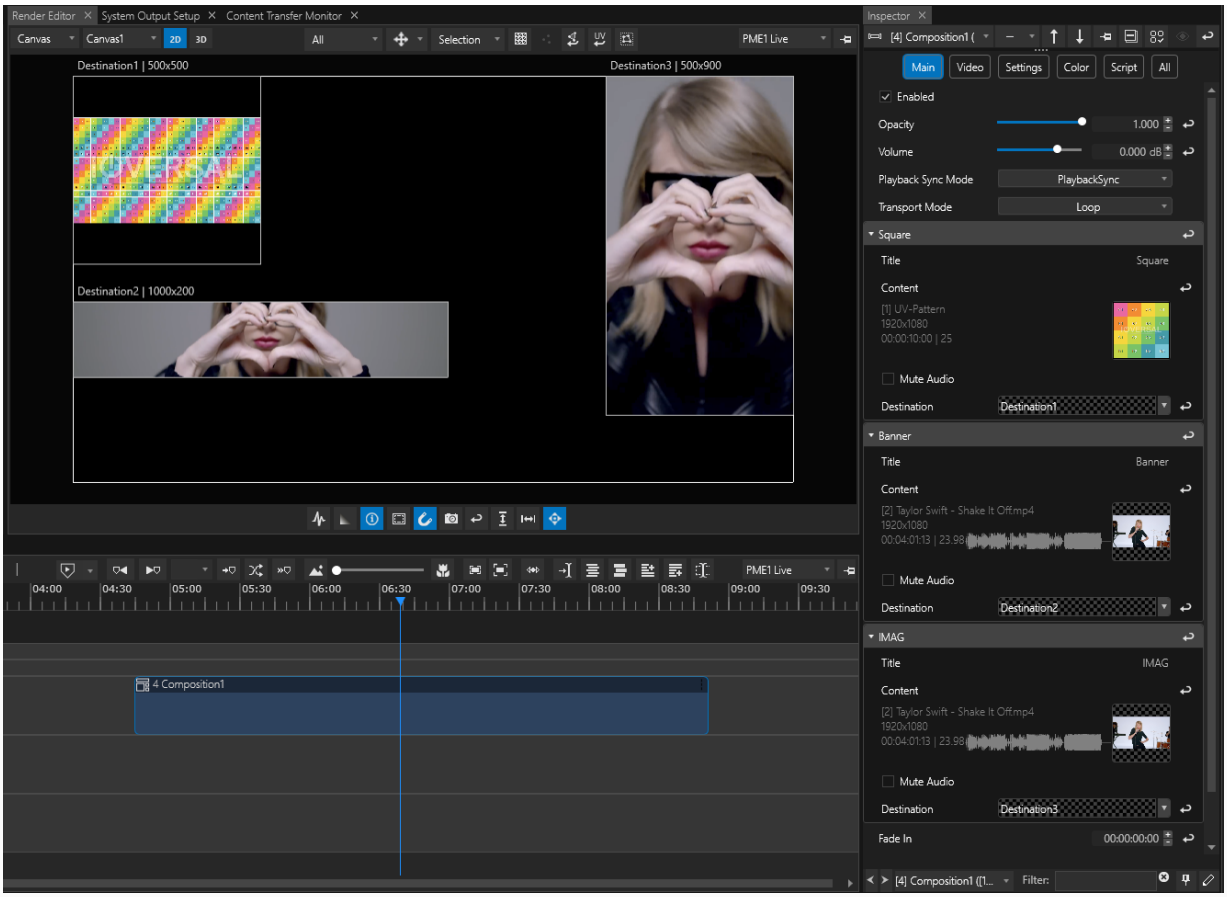
Reusable Structures - Think of Compositions like "master slides" in presentation software. They define structure and behavior, not the actual media.

Integrated Transitions - Base functions such as fade-in and fade-out can be set at the Composition level and are automatically applied to all clips used with it.

Rapid Workflow - Create Compositions directly from selected Destinations, or set a default Composition for a track. Dragging multiple clips onto that track will automatically create a Composition Clip and fill the slots of the Composition.

Scalability - Especially useful for installations with many outputs (e.g. stadiums, stage setups, advertising playback), where repetitive multi-screen timelines can be assembled quickly without cluttering tracks.

Compositions streamline multi-output workflows, reduce timeline complexity, and improve performance by consolidating repeated structures into efficient reusable layouts.



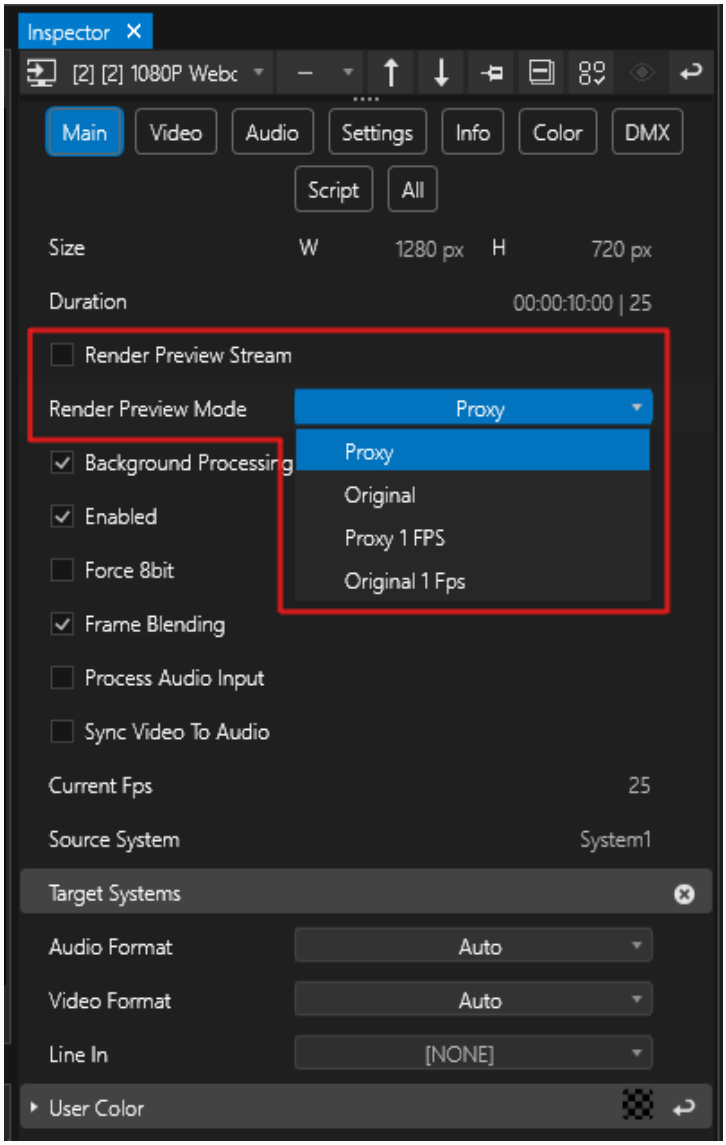
Both Destinations and Compositions can, of course, be used not only in Sequences but also in Playlists and Shotboxes.

Live Input Preview in Remote Render Editor

Live Inputs (HDMI, SDI, etc.) that are available locally on a Session Member can now be streamed over the network to all other Session Members for preview purposes, allowing them to be displayed in the Render Editor. The **Render Preview Stream** must be activated individually for each Live Input content. Currently, four different modes are available:

- **Proxy** (lower resolution, original FPS)
- **Original** (original resolution, original FPS)
- **Proxy 1 FPS** (lower resolution, 1 frame per second)
- **Original 1 FPS** (original resolution, 1 frame per second)

For reference, streaming a "Original" Full HD signal at 50 FPS requires approximately 100 Mbit/s per stream and per receiver. Providing the stream impacts the server's performance, so this feature should be used with care in live show situations.

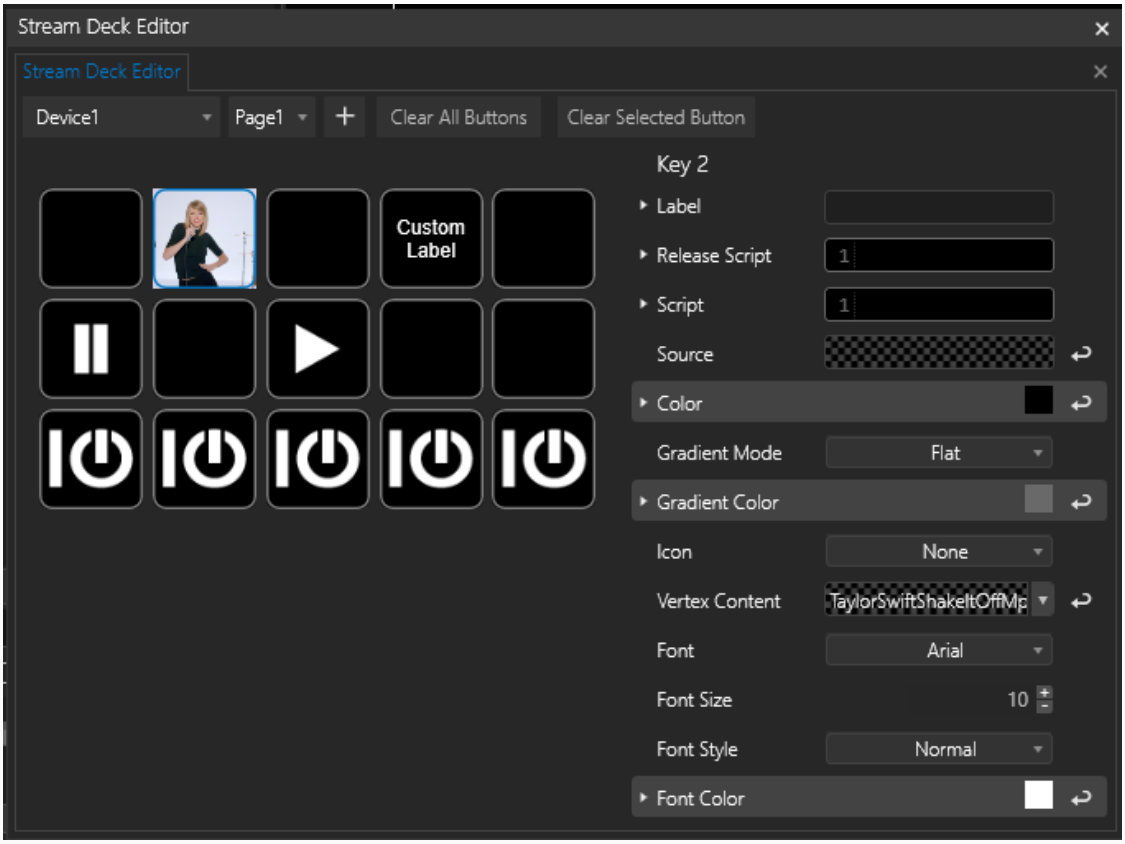


Stream Deck Editor

Elgato Stream Decks can now be configured even more conveniently for use with Vertex. In the **WINDOWS** menu (submenu *Tools*), there is a new **Stream Deck Editor**. Stream Deck devices that were previously added from the library to the project can be edited here using a graphical representation.

Buttons can also be copied to other buttons via drag & drop, and tested by holding down the **CTRL** key and clicking with the mouse.

In addition, subpages for Stream Decks can be created and managed in this way. For navigation, scripts such as `Device1.GotoPage 2` or `Device1.GotoNextPage` can be used.

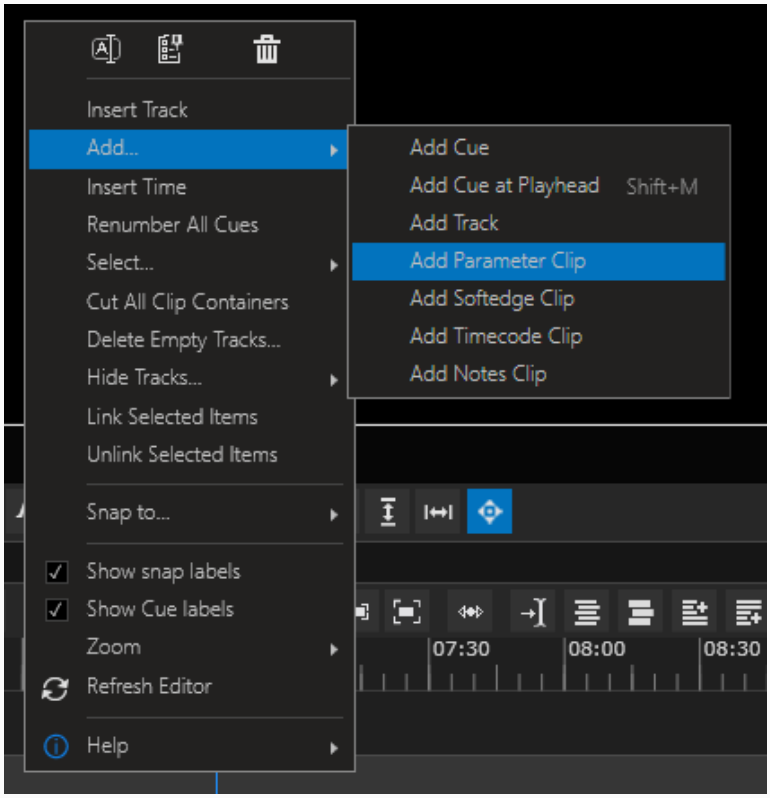


Project Bundle

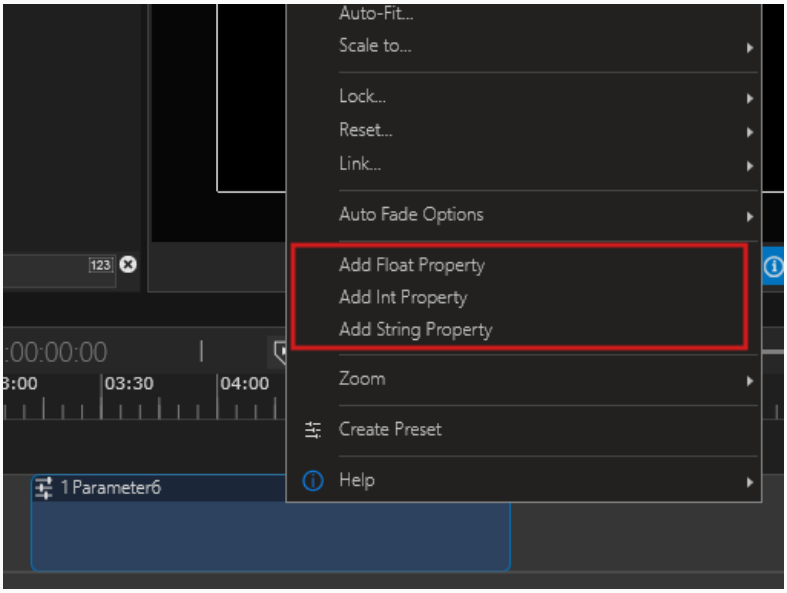
Managing the contents belonging to a project can quickly become a tricky task in a network of multiple Vertex systems. While contents can potentially be imported by all session members, they are not necessarily transferred to all other members. In addition, there are many possible locations for both the source files and the transferred files.
To consolidate all project contents in one central place - for example, to fully archive a project - we have implemented a tool that guides the user step by step through the process of creating a Project Bundle. This tool can be found in the TOOLS menu, under the "Content Cleanup" submenu, and is called **Bundle All Content to Local System.**

Parameter Clips

In more complex projects, there may be a requirement to dynamically change values in sync with playback. These values can, for example, be sent to external components via e.g UDP, MIDI or MQTT, or applied internally in Vertex after being processed, for instance, through a node system.

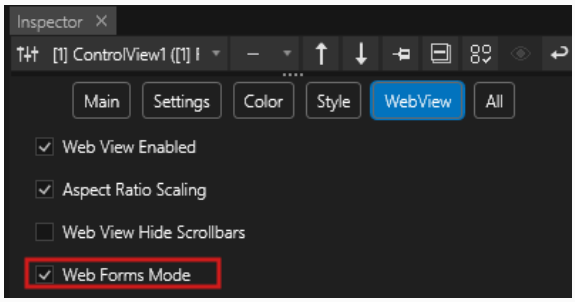


For this purpose, we have introduced the so-called **Parameter Clips**. These can be created by right-clicking in an empty area of a track in the Sequence Editor. By right clicking the created Parameter Clip, you can add any number of Float, Int, or String parameters. These parameters can then be manipulated in the sequence via keyframe animation and, as usual, dragged & dropped from the inspector into other editors for further processing.



Control View: Web Forms Mode

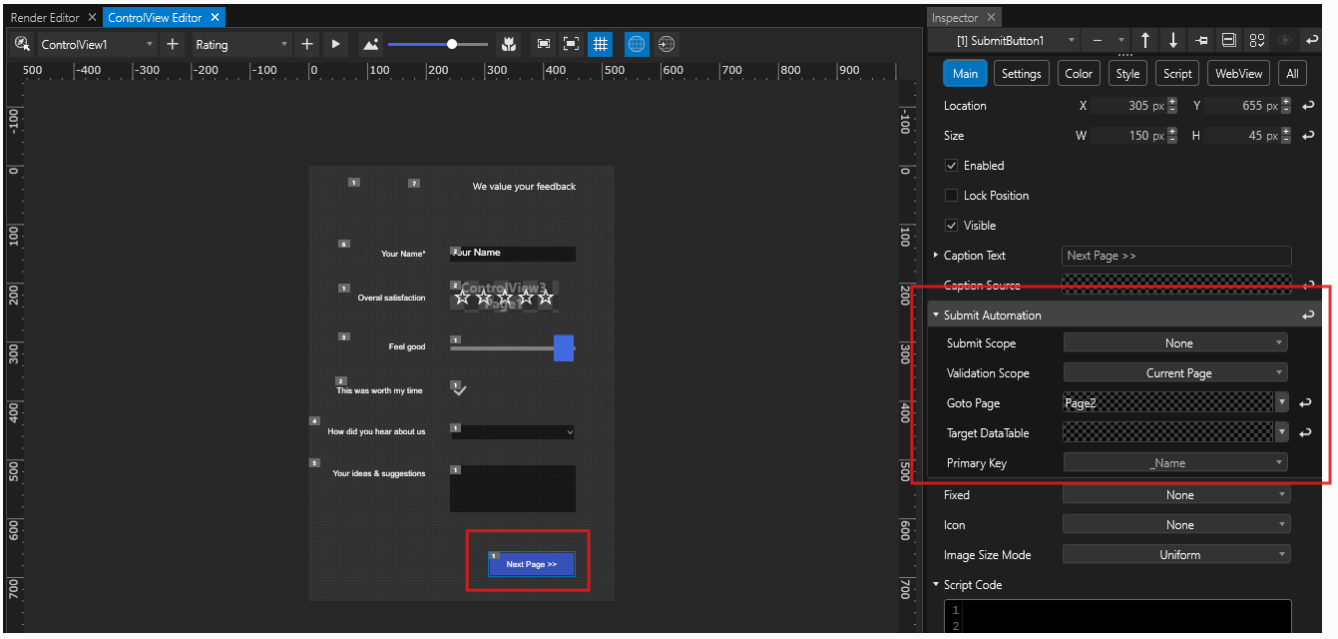
Control Views now support a new **Web Forms Mode**, enabling the creation of independent web-based forms. Unlike the standard synchronized behavior, controls displayed in multiple Web Views no longer mirror each other once Web Forms Mode is activated.



Web Forms Mode: Activate this option in a Control View to decouple controls from synchronization across multiple Web Views. Each browser instance will then handle its own form state independently.

All Controls can be added to pages as usual and are fully functional within the form workflow.

Submit Button: A new control type has been introduced with extended functionality compared to the standard Click Button.

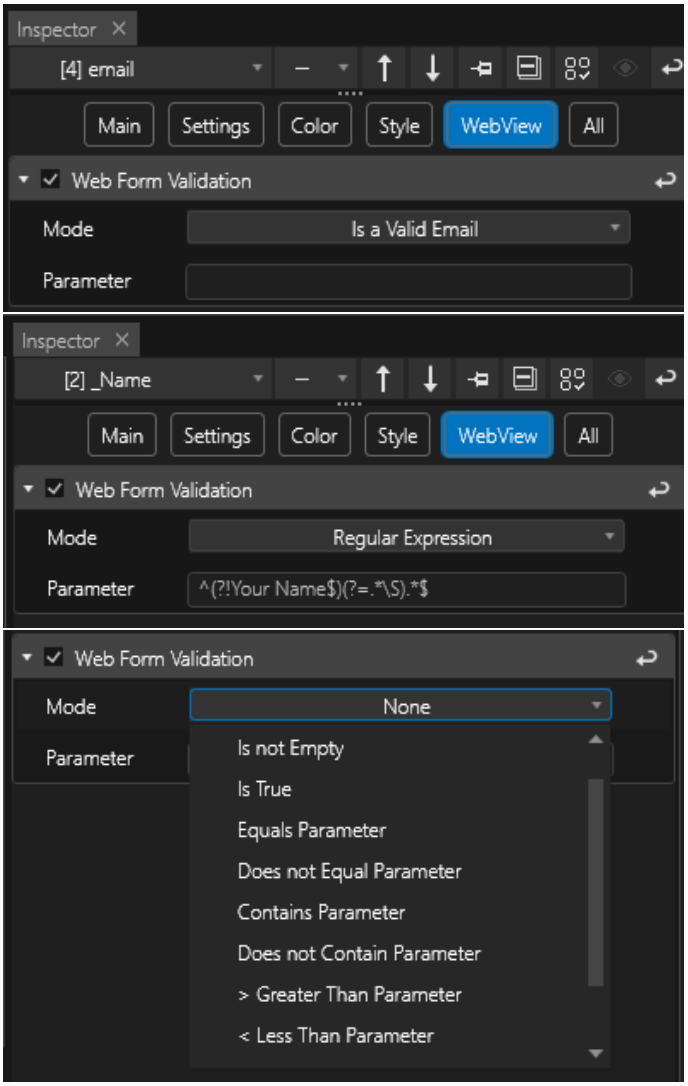


The Parent Property "Submit Automation" comes with the following settings:

- **Submit Scope** - select which Controls shall written into the DataTable:
  none, Current Page's Controls or All Control Views Controls
- **Validation Scope** - determines whether validation is applied to the current page only or to all controls in the Control View.
- **Goto Page** - target page to redirect the Web Client to - but only if validation is successful
- **Target DataTable** - Data can be written into a DataTable object for further evaluation
- **Primary Key** - Inspired by database models, a Primary Key can be selected here, which serves as the entry identifier in the DataTable. All Controls that are available in the Control View or on the Page (depending on the selection of "Submit Scope") are automatically offered as options.

Each individual Control provides its own **validation logic**. For example:

- A *Text Input* field for "Your Name" can validate input against a regular expression.
- An *Email* field automatically checks for valid email formatting.



If all validations succeed, the user is redirected to the defined page, and the values of the controls are written to the specified data table.

➪ Check out the new **Sample Project** "ControlView Web Forms" to get in touch with this feature!

Control View: receive IP address of current Web Client

It is often important to determine from which IP address the last interaction in the Web View of a Control View was executed.
For this purpose, the script code of the button can now access this information via `Context.Ip`.

Example:

Script code of a ClickButton in a ControlView:

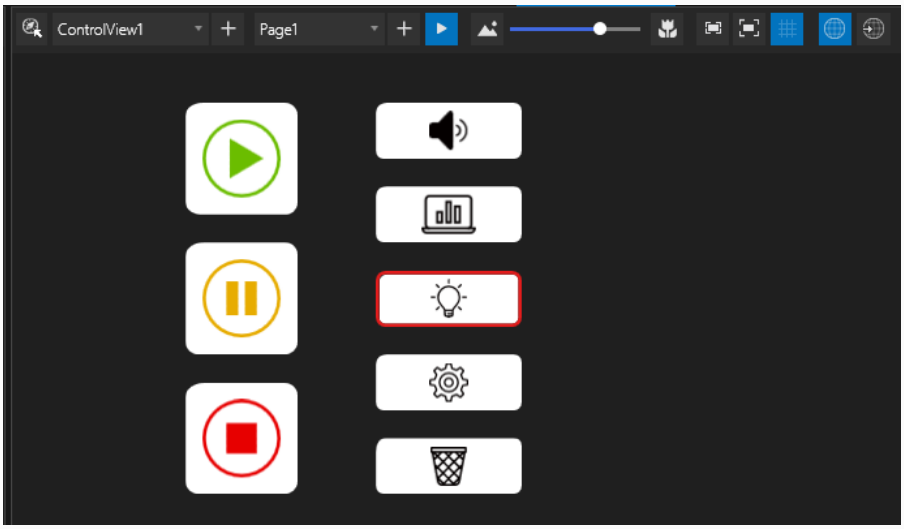
```
Variable1 = Context.Ip
```

Result:

When this button is clicked in the Web View, the IP address of the web client will be stored in Variable1.

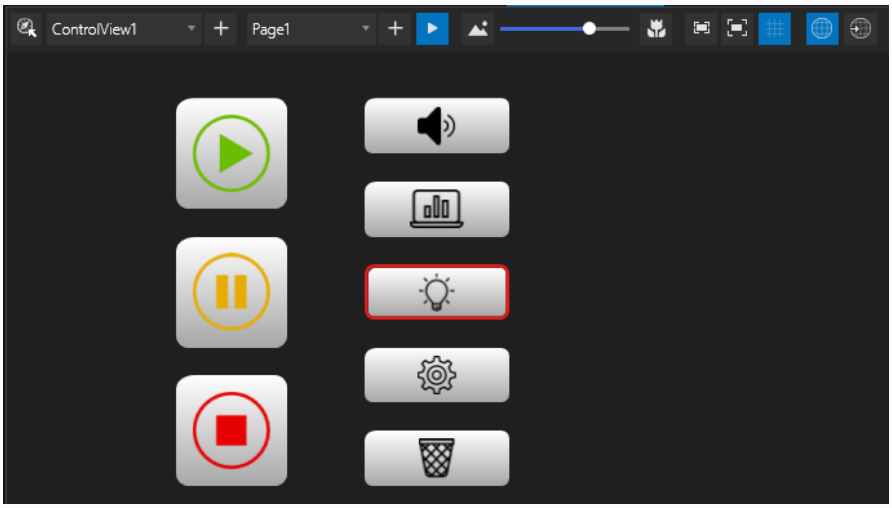
Control View: Button Icons

Buttons now have access to a wide range of predefined icons. Icons can either be assigned to a button via drag & drop from the "Icon" library or selected directly within the button using a dropdown parameter. The icons can be colored independently and, being vector graphics, automatically scale with the button to any size.



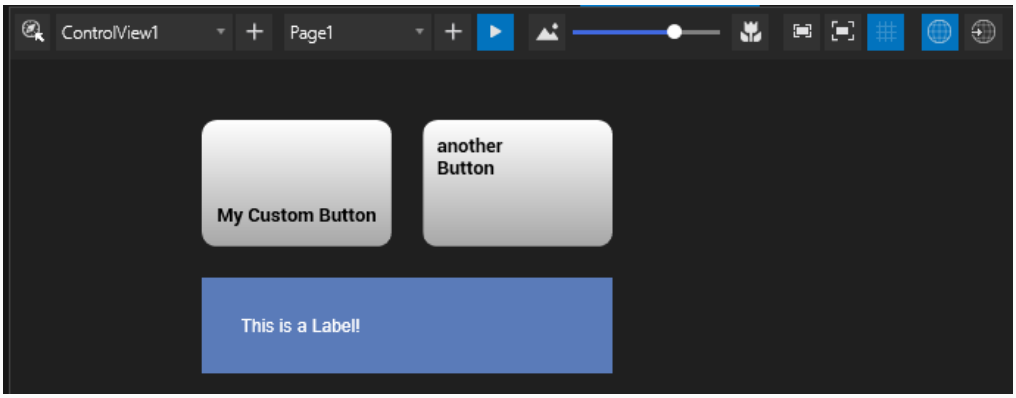
Control View: Button Gradients

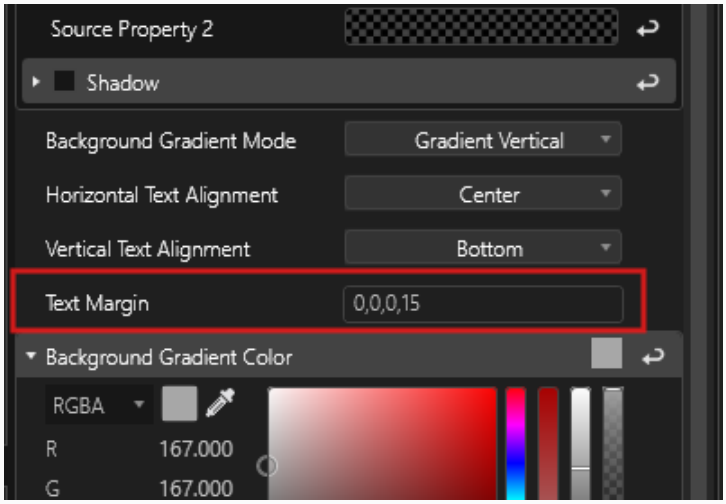
Buttons can be assigned predefined or fully customizable background gradients.



Control View: Controls Margin

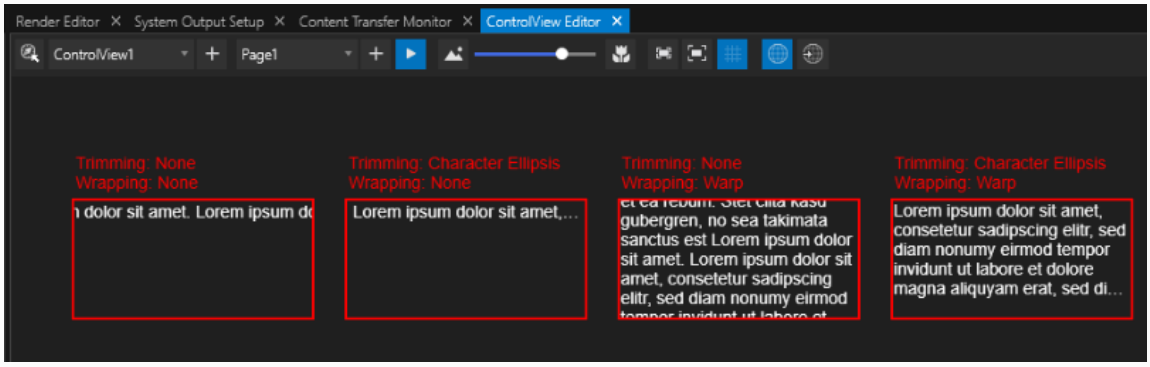
We have added the ability for buttons and labels to adjust text positioning within the control using individually configurable margins. The margin is applied relative to the selected alignment and can either be a single value, affecting the corresponding edge(s) for left-, right-, top-, or bottom-aligned text, or it can be defined absolutely for all four edges (left, top, right, bottom).





Control View: Label Trimming and Wrapping

Label controls in the Control View now include two additional properties. These allow the label text to be automatically trimmed so that it ends with "..." at the outer edge of the label if the text does not fully fit within the field. In addition, line breaks can be applied automatically. Both properties can be used in combination. (Currently only available in UI, not in Web View)

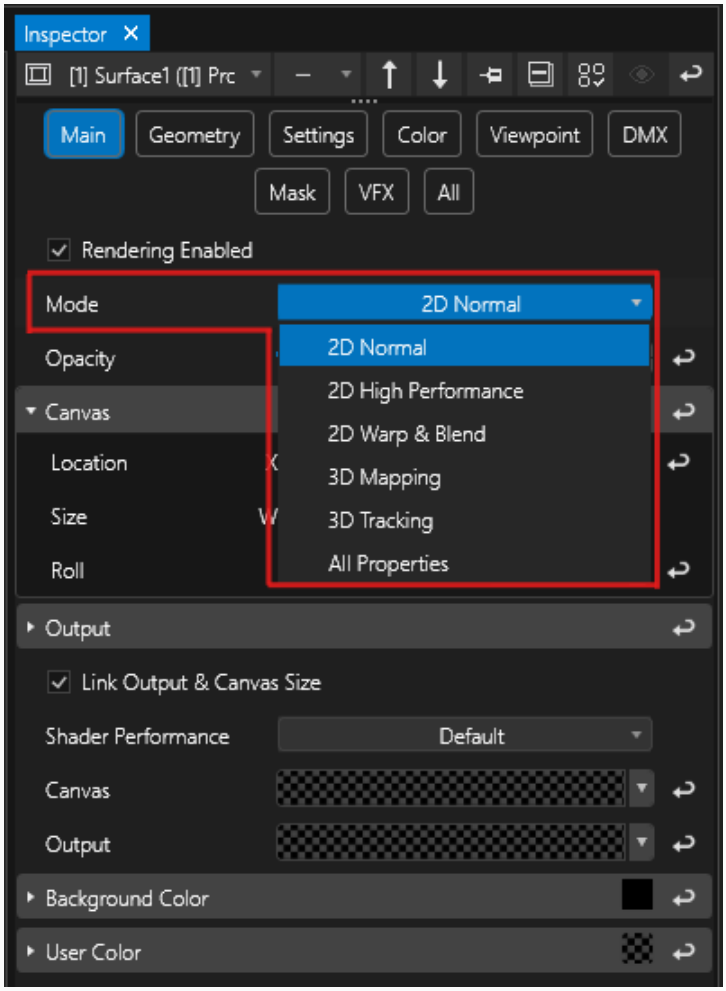


MIDI TimeCode Out

Vertex can now send MIDI Timecode based on the timecode of a selected playback. To do so, first add a **MIDI TC Sender** from the Device Library. In the properties of the playback whose timecode should be sent via the selected MIDI device, you can then activate the property **MIDI Time Code Sender Enabled** in the *Timecode* category of the Inspector and assign the corresponding device to the property **MIDI Time Code Sender** via drag & drop.

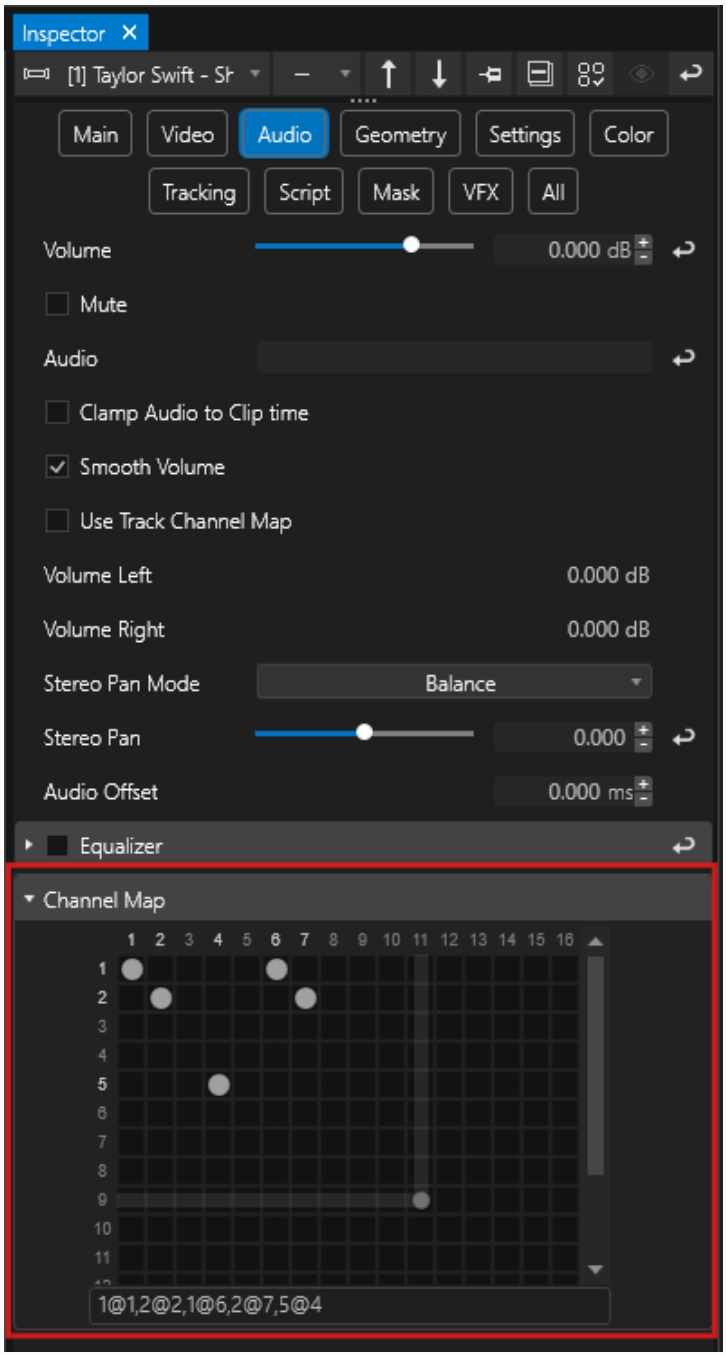
Surface Modes (property filtering)

Surface objects are complex in their functionality and therefore come with a large number of properties. However, many of these properties are only needed in specific use cases and unnecessarily overload the inspector. For this reason, we have introduced **Surface Modes,** which filter the properties shown in the inspector depending on the use case. (Every property still exists and is applied regardless of the selected mode - they are simply shown or hidden!



Audio Channel Map Matrix

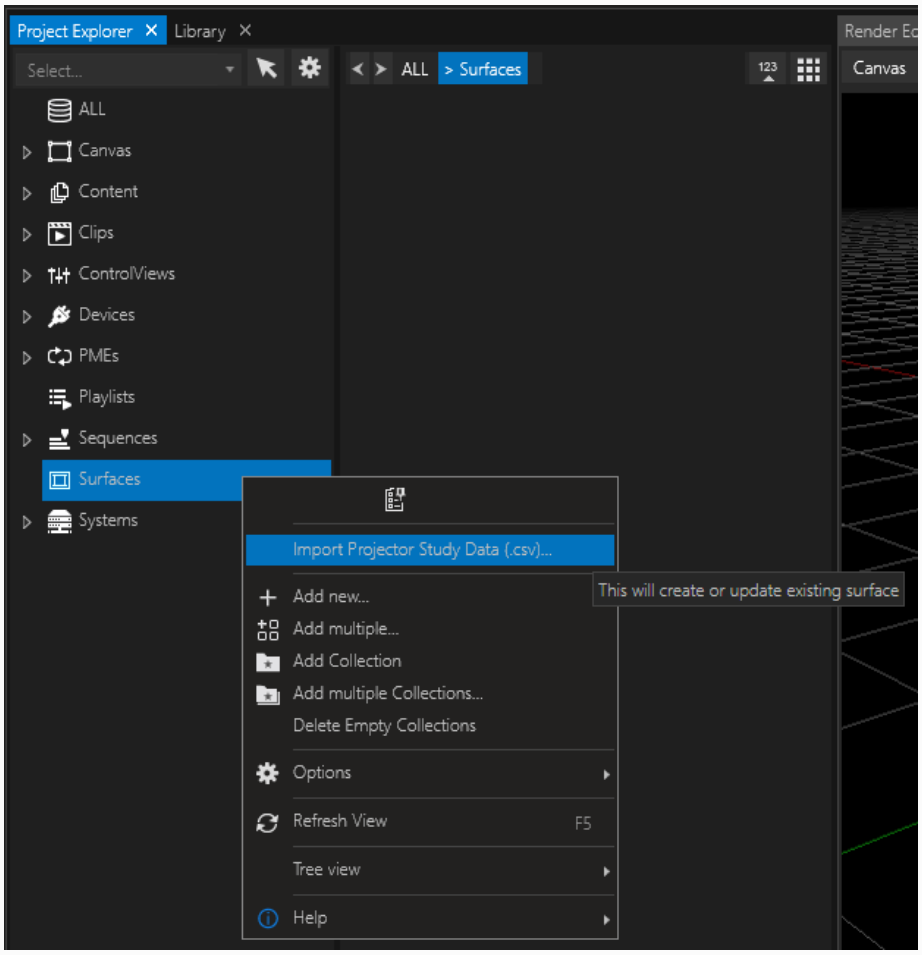
Clip Containers in Vertex have always been able to receive an individual audio channel map, allowing the clip's audio channels to be routed to specific output channels in Vertex. The command line-based mapping has now been enhanced with a visual **Audio Channel Map Matrix**, which not only represents the command line input but also allows direct changes to the mapping by clicking on the nodes.



Projector Study Data Import

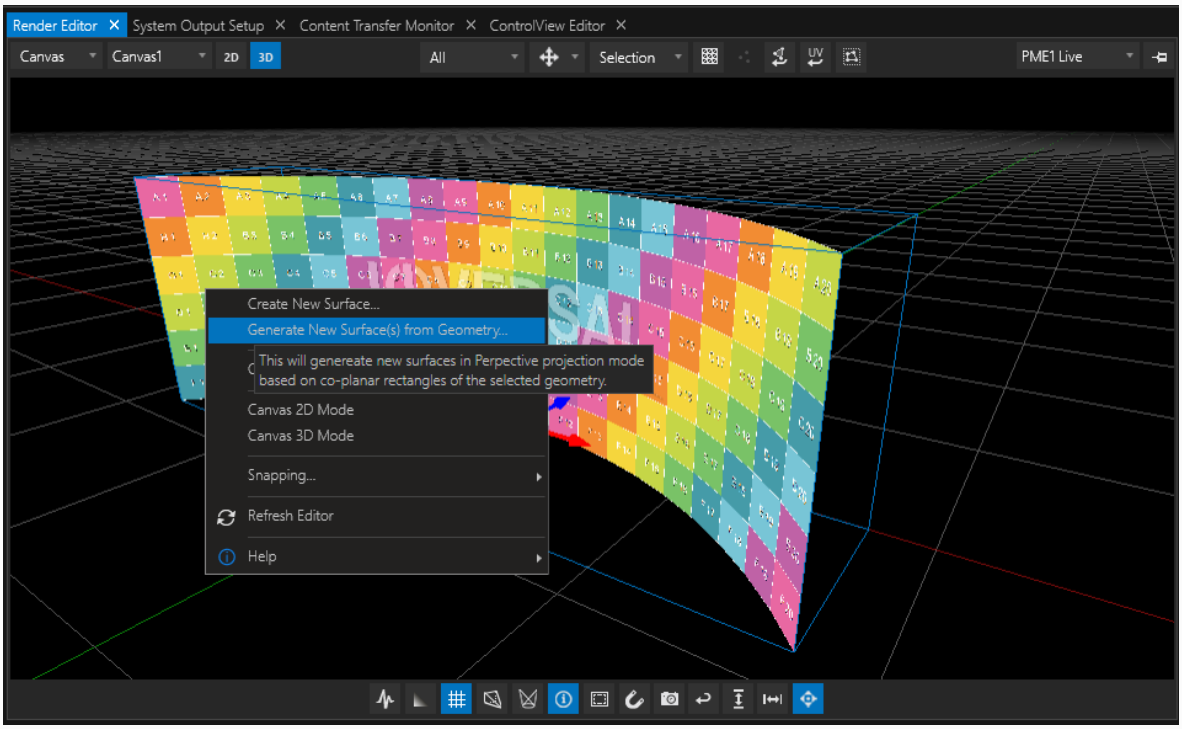
Projector studies created in an external tool can now be imported into Vertex. This process generates surfaces in Vertex whose parameters exactly match those defined in the projector study and exported as a CSV file. Positions and orientations of Surfaces in the Vertex 3D environment as well as viewpoint parameters will correspond the projector data.

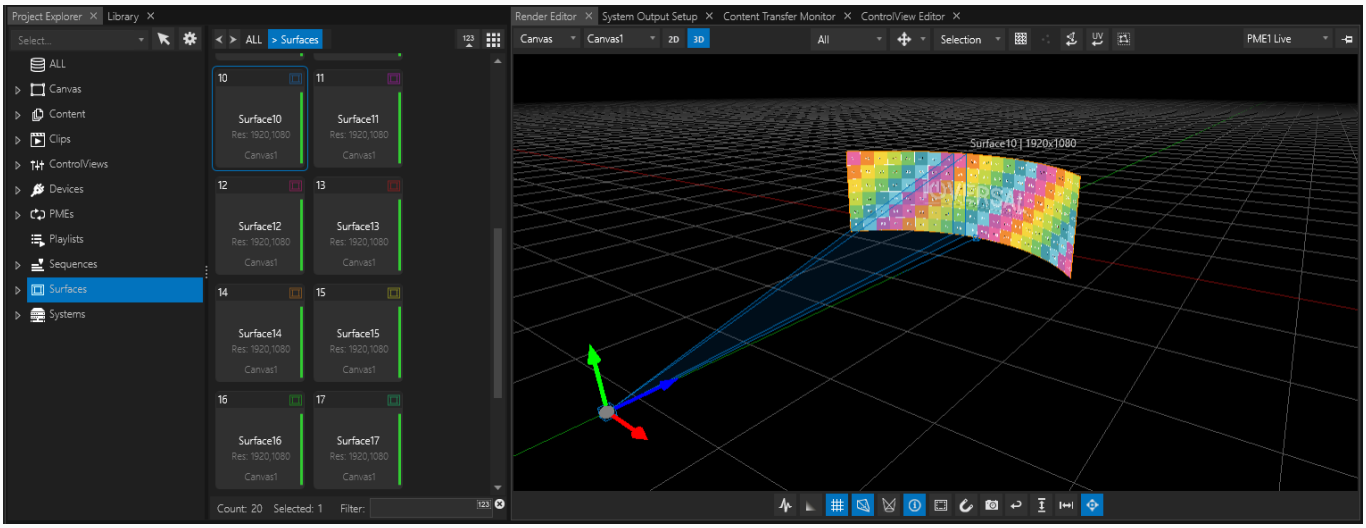
When re-importing the projector study, the previously created surfaces are updated instead of new ones being generated.



Auto-Generate Surfaces from Geometry

In XR studio environments with tracked cameras and "projected" playback of content onto a segmented, curved LED wall from the camera's axis, it is necessary in Vertex to create a separate surface for each segment of the curved LED wall. The creation of this surface array can now be automated, provided that the geometry of the LED screen exists in Vertex in the form of a 3D model.





Logitech MX Creative Console Device

Use the Logitech MX Creative Console, consisting of a keypad and dialpad, in combination with Vertex. The corresponding plugin for Vertex is available for free download in the Logitech Plugin Marketplace.

Shotbox “All Solo” mode

The playback provider *Shotbox* can now be operated in the so-called **All Solo** mode. This mode can be enabled or disabled via the toolbar in the Playback Editor.
If the **All Solo** function is activated during ongoing playback of one or more items, these items will continue to play in their current mode until the next item is triggered. Starting the new item causes all previously active items to be deactivated/hidden, ensuring that only the new item is rendered. Each subsequently activated item will then also be exclusively active.

Cue “Fade To” Modes

In addition to the already familiar cue modes such as GotoCue or GotoTime, it is now possible to select modes that perform a fade to the desired target (CueID, Timecode, Next Cue, Previous Cue) within a configurable preload time and fade time.

Cue “Run Script on Play”

The previous behavior of cues in “Pause” mode was that the Script Code linked to the cue was executed at the moment the playhead entered the pause cue. With the optional property **Run Script on Play**, it is now possible to execute the script when the cue is exited by resuming playback (“Play”).

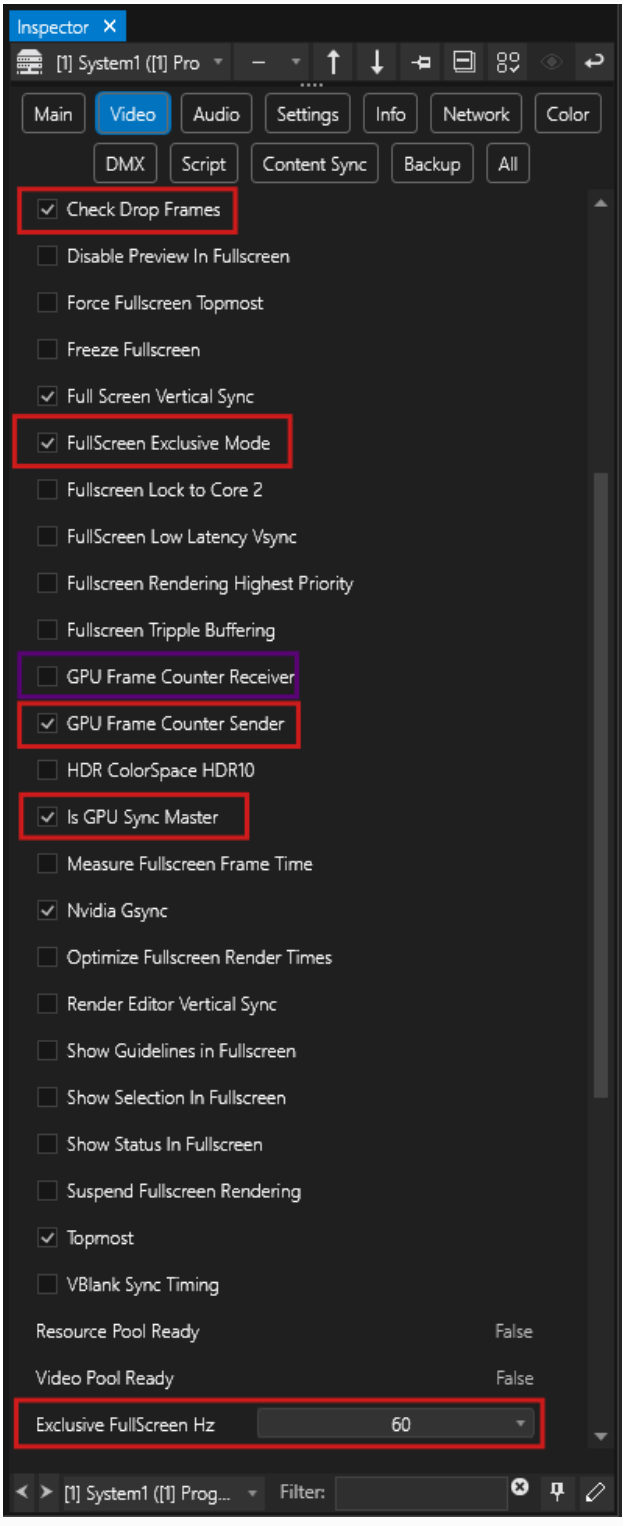
Advanced Multi-System GPU Sync

Vertex natively provides the ability to synchronize multiple systems so that all systems reliably render the same frame of the same content.

For true frame-build synchronization - which is essential, for example, when driving very large LED screens across multiple systems - it is necessary to synchronize the GPUs on a hardware level. Without this synchronization, transitions between systems would otherwise show visible tearing due to slight delays in image build-up.

By combining both technologies - Vertex's internal synchronization and GPU-level synchronization - we have implemented additional methods in Vertex. These compensate for instabilities in rendering (such as frame drops) and optimize unreliable synchronization information passed through the NVIDIA API to Vertex.

When playing back content in *Display Rate Mode*, we recommend applying the following settings on playout systems in a GPU-synchronized multi-system setup:



FullScreen Exclusive Mode ON (bypasses Windows DWM - requires a restart of Vertex. The Full Screen Renderer must cover the system's entire virtual desktop)

Exclusive FullScreen Hz set to incoming genlock FPS

Check Drop Frames ON (detects and corrects dropped frames)

Define one render system as both **GPU Frame Counter Sender** and **Is GPU Sync Master**

Set all other playout systems to **GPU Frame Counter Receiver**, keep both parameters from previous line deactivated

Once these settings are applied, fully synchronized clip rendering is ensured. This also applies when using the playback sync mode **FreeSync.**

Contents split into multiple slices (for example, when resolution exceeds the maximum supported size) can be synchronized across systems by linking the corresponding clip containers in the sequence. To do this, select the containers containing the slices, right-click, and choose *Link...* → *Link selected items*. In this way, all slices of large content remain synchronized across systems.

It is important that GPUs are clocked using an external sync generator and that the NVIDIA sync trigger is set to *falling edges*. The FPS of the sync generator must exactly match both the FPS configured on the GPU outputs and the FPS of the content being processed.

Various Scripts added for content management

Based on user requests, as Vertex is increasingly being controlled remotely both through external content management systems and within management environments developed in the Vertex Control View, we have implemented additional scripts. These scripts make it easier to manage contents and collections and also allow information to be returned from collections or playlists, ensuring that feedback from Vertex about available elements is accurately represented in the CMS. Sample Scripts:

`ContentManager.Collection1.AddNewContent` (Parameters: *SourceSystem,FilePath*)
`ContentManager.Collection1.ListAllItems` (returns array of Items)
`ContentManager.AddCollectionAsId` (returns ID of newly created Collection)
`Playlist1.ListTracks` (returns json of all Playlist tracks with multi information)
`Playlist1.Tracks.Move` (Parameters*: IndexA, IndexB*, moves TrackA to IndexB)
`PlaylistManager.AddNewAsPlaybackId` (returns ID of Playback assigned to newly created Playlist)

ClipContainer Enter/Leave Script added

Clip Containers offer a variety of options for automatically executing scripts in response to different events. The following overview outlines the available script types:

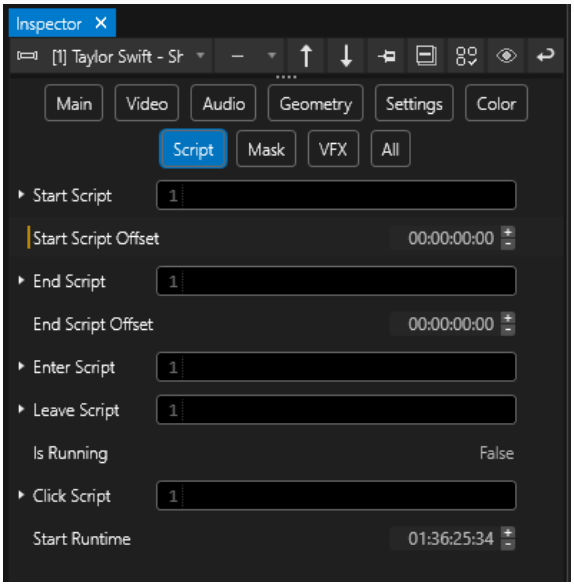
Start Script: Executed the moment the playhead enters the container. This timing can optionally be adjusted using the *Start Script Offset* property.

End Script: Executed the moment the playhead exits the container. This timing can optionally be adjusted using the *End Script Offset* property.

Enter Script: Executed whenever the playhead enters the container - regardless of position. For example, it will also trigger if the playhead jumps directly into the middle of the container.

Leave Script: Executed whenever the playhead leaves the container - regardless of position. This means it will also trigger if the container is exited mid-play, e.g., manually, via a cue, or via script.

Click Script: Works in combination with Full Screen Interaction. If the object rendered by this container is clicked (mouse or touch) in the Full Screen Renderer, this script will be executed. This enables the quick and easy creation of interactive control interfaces directly in the Vertex renderer.



Additionally, two parameters provide useful runtime information:

Is Running: Indicates whether the container is currently active.

Start Runtime: Displays when the container was last started, measured against the runtime of the master system.

“Frame Time” in System Fullscreen Status

In addition to the quantity of frames the system is rendering per second (and the time every frame is displayed), we have added a very valuable parameter to the Fullscreen Status. The Frame Time reports the time it takes the Vertex render engine to process each render frame in both milliseconds as well as percentage of the overall frame. This way a user can easily identify and troubleshoot performance bottlenecks and become aware of available headroom.