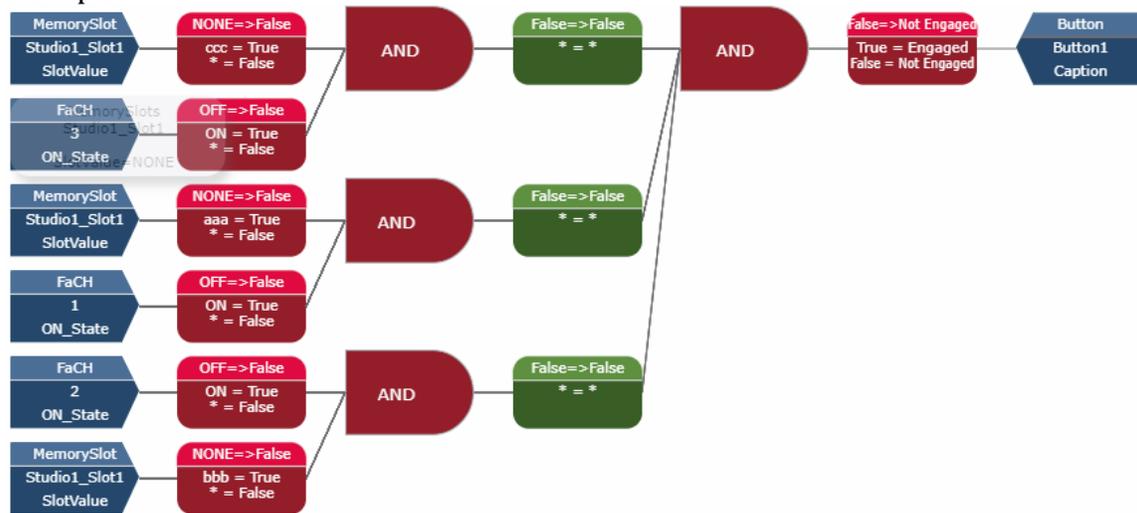


Version 1.5.0.01 Changes

Recursion Detection

This version introduces recursion detection into Logic Flow translators. If the Logic Flow system detects a given translator being activated more than 20 times in 1000 milliseconds, it will assume there is recursion involved and disable the translator to prevent excess cpu load. In some situations, this can create a false positive. For example:



In this flow the same memory slot is used for multiple entry points to a flow so a change to that memory slot could cause multiple pass throughs of some translators. If this flow were to grow to a much larger variation with many more entry points of the same memory slot, it could be possible to trip the recursion detection incorrectly. There are probably better ways to craft this flow but in the case of a false positive, the values may be increased using the API on port 9600. See the Pathfinder manual for more information on using the API. For example:

```
Set LogicFlows#0.LogicFlowFolder#Test.ListTranslator#6ef05ca8-e530-4326-8a35-85e9e41495af ExecutionLimit="50/1000"
```

This command would increase the recursion detection limits on the specified translator to 50 times per 1000 milliseconds.

If a translator gets disabled via the recursion detection, a log message will be generated and an attempt to send an email message to the critical event email message if it is configured will be made.

There is no UI for adjusting recursion detection parameters in this version though that is planned for a future version.

Version 1.5.1.02 Changes

Generic Emulator

Initialization Message

This version adds the ability to configure an initialization message that will be sent when a generic emulator connects. This allows the emulator to automatically send login or other initialization messages.

Device Emulator Editor

Emulator Name:

Emulator Type:

Connection Settings

Connection Type:

Port:

Generic Emulator Watchers

Init Message:

The same escape characters may be used as for the ToSend property of the Device Emulators. These include:

- \cr = Carriage Return
- \lf = Line Feed
- \t = Tab
- \%XX = hexadecimal two digit ascii character.
- Double slash to escape the escape slash and send the literal value.

In the case of listener types, this message will be sent to each client that connects to the listener. In the case of client connection types, this message will be sent each time the client connects.

Additional Emulator properties for Logic Flows

This version also adds some additional properties related to Generic Emulators that may be used in Flows.

- Connected – This property will be true if there are any clients connected or if the tcp client is successfully connected and false otherwise.
- ConnectedCount – This property will retain the number of currently connected clients to the emulator.
- ConnectionLost – This property will switch to True and then back to False when a client disconnects.
- ConnectionObtained – This property will switch to True and then back to False each time a client connects.

Version 1.5.3.03 Changes

Infinity

This version adds some basic support for Infinity products. A device that is detected as an infinity product has a `SendToRestApi` property which can be used by logic flows to address the Infinity Rest API. The format of the commands addressed to this property should be:

- Operator Path Value
- For example:
 - `PUT /api/s/main/audio/speakerMute true`

Note that gets are not working at this point in time as we do not want to encourage polling logic. It is also important to know if the value being sent needs to be wrapped in quote. Contact support for details on properties that may be manipulated using the Infinity Rest API.

Memory Slots

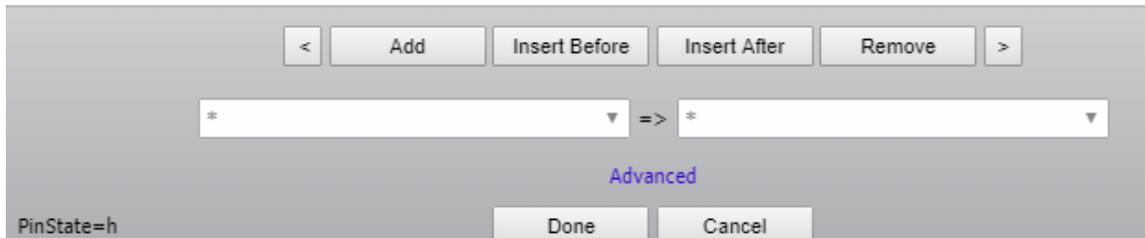
This version adds two properties to the main memory slot manager.

- `ChangeAllByValue` – This property is a write only property that will change all memory slots that have a certain value to another value. In logic flows it is only available in the API tree. The syntax is:
 - `oldvalue-->newvalue`
 - For example:
 - `Myvalue-->MyNewValue`
 - This would change every memory slot in the system that has a value of `MyValue` to the value `MyNewValue`
- `CopyTo` – This property would copy the value of a memory slot to another memory slot. In logic flows it is only available in the API tree. The syntax is:
 - `sourceslot-->targetslot`
 - For example:
 - `MySlot-->YourSlot`
 - This would update the value in `YourSlot` with whatever the value is in `MySlot`.

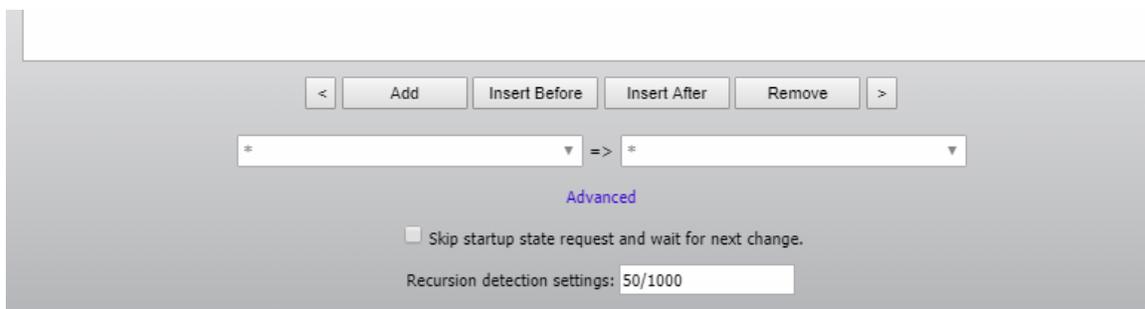
Version 1.5.5.05 Changes

Logic Flows Recursion Detection User Interface

This version adds a user interface for configuring the recursion detection settings on a translator. Editing a translator now presents an Advanced link.



The startup state checkbox has been moved under this advanced link as well as a box for the recursion detection settings. Clicking the Advanced link will show or hide these settings.



50/1000 represents that the recursion detection will disable the translator if the translator is being analyzed more than 50 times per 1000 milliseconds.

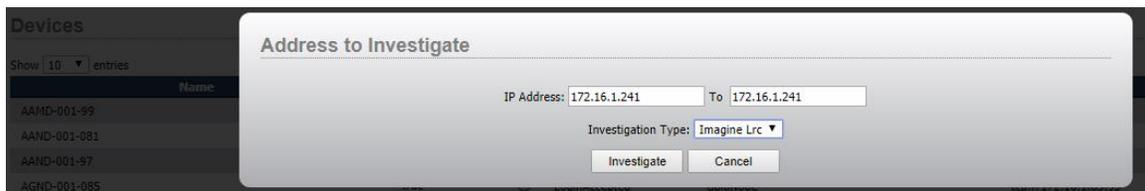
Additionally, there is an API property that should be used with extreme care that can change the default recursion settings for all new translators. It will also update any existing translators that are currently at the default settings. This can be cpu intensive and require a significant amount of writing to disk. Please make a backup before changing this value. It is available via the API:

- Set LogicFlows#0 DefaultRecursionSettings="50/1000"

Version 1.5.8.07 Changes

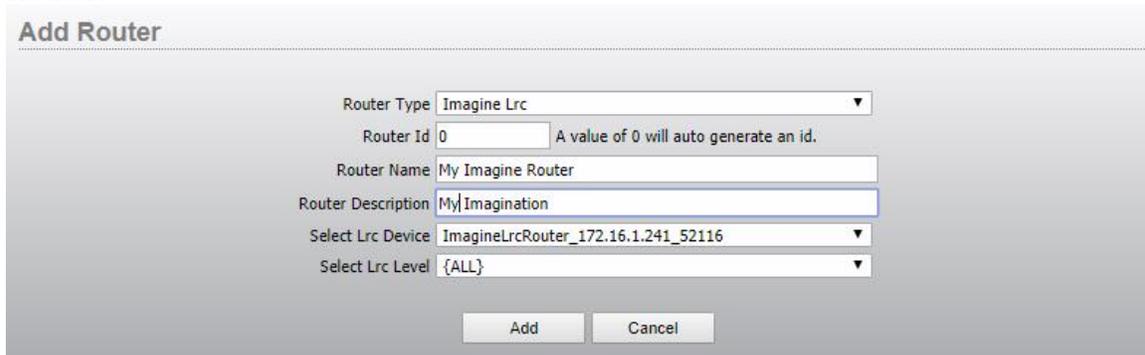
Imagine Logical Router Control

Pathfinder Core PRO has added support for Imagine Logical Router Control protocol for controlling Imagine Routers. To add an Imagine Router into Pathfinder Core PRO, you first must add the Imagine Device to the Devices list and then you can add a router. In Devices, click the Add button, enter the IP address and select Imagine LRC as the investigation type.



The screenshot shows a modal dialog titled "Address to Investigate". On the left, a "Devices" sidebar is partially visible with a list of device names: AAAND-001-99, AAAND-001-091, AAAND-001-97, and AGND-001-085. The main dialog area contains two input fields for "IP Address" and "To", both containing the value "172.16.1.241". Below these is a dropdown menu for "Investigation Type" set to "Imagine Lrc". At the bottom of the dialog are "Investigate" and "Cancel" buttons.

Once the device has been successfully added, click on the Routers tab and add a new Router.



The screenshot shows the "Add Router" form. It has several fields: "Router Type" is a dropdown menu set to "Imagine Lrc"; "Router Id" is a text input with "0" and a note "A value of 0 will auto generate an id."; "Router Name" is a text input with "My Imagine Router"; "Router Description" is a text input with "My Imagination"; "Select Lrc Device" is a dropdown menu set to "ImagineLrcRouter_172.16.1.241_52116"; "Select Lrc Level" is a dropdown menu set to "{ALL}". At the bottom are "Add" and "Cancel" buttons.

Select the Imagine Lrc type from the Router Type drop down. Give the Router a name and a description (if desired). Select the Device that was added and select the router level you wish to add.

Imagine routers have multiple levels and PathfinderCore PRO can control full routes which route all levels at once by adding the {ALL} level or any given breakaway level may be added as a separate level. Or both may be added in different routers. This allows routes that should target all levels to be made on the router with the All target and routes of individual break away levels to be made on the breakaway routers.

Once added the router will discover the IOs made available by the selected level and routes may be manipulated in the same way as other routers in the system

Source		Destination	
Name	Description	Name	Description
SRC 13	SRC 13 ON ImagineLrcRouter	DST 1	DST 1 ON ImagineLrcRouter
		DST 11	DST 11 ON ImagineLrcRouter
SRC 13	SRC 13 ON ImagineLrcRouter	DST 12	DST 12 ON ImagineLrcRouter
SRC 16	SRC 16 ON ImagineLrcRouter	DST 14	DST 14 ON ImagineLrcRouter
		DST 15	DST 15 ON ImagineLrcRouter
		DST 2	DST 2 ON ImagineLrcRouter
SRC 13	SRC 13 ON ImagineLrcRouter	DST 3	DST 3 ON ImagineLrcRouter
		DST 4	DST 4 ON ImagineLrcRouter
		DST 5	DST 5 ON ImagineLrcRouter
SRC 16	SRC 16 ON ImagineLrcRouter	DST 6	DST 6 ON ImagineLrcRouter

Important Notes: Support of Imagine routers is very new. It should be very much treated as a beta feature. Also lock states do not currently reflect control locking in the Imagine router. That may be added as a future feature.

Lwrp STAT Properties

This version includes data obtained from the STAT command in Lwrp for devices that support it. These read-only properties may be found in the API tree in logic flows under the device branch that supports it.

```

  ▸ Src#1 SRC 1
  ▸ Src#5 SRC 5
  ▸ Src#6 SRC 6
  ▸ Src#7 SRC 7
  ▸ Src#8 SRC 8
  ▾ Stat
    ● FriendlyName
    ● SapId
    ▾ StatDst#1
      ● ReceiveAddress
      ● Stream
    ▸ StatDst#2
    ▸ StatDst#3
    ▸ StatDst#4
    ▸ StatDst#5
    ▸ StatDst#6
  
```

There are four types of STAT objects described below:

STAT SRC

- StatSourceAddress = Source Address data.

STAT DST

- Stream = Whether the stream assigned to this destination is up or down.
- Receive Address = Receive address data often used with Aes67 indications.

STAT ICH

- AesSync = Only appears on AES inputs and displays whether AES sync is in error or ok.

STAT SYNC

- Master – master synchronization data.
- Slave – slave synchronization data.
- Sync – Synchronization source.
- FrequencyAdjust – adjustment value in ppm.

Important Note: Stat Sync data collection is not enabled by default. That is because it can generate a lot of data especially in systems with large numbers of X nodes. To enable stat sync data collection for a given device, use the api on port 9600 and send the command:

```
SET Devices#0.XNodeCombo#[tcp://172.16.1.95:93].LwrpInterpreter#0  
SubscribeToSync=True
```

Also note that while turning the SubscribeToSync option on for a device will be saved between restarts, the state is not currently cluster synchronized.

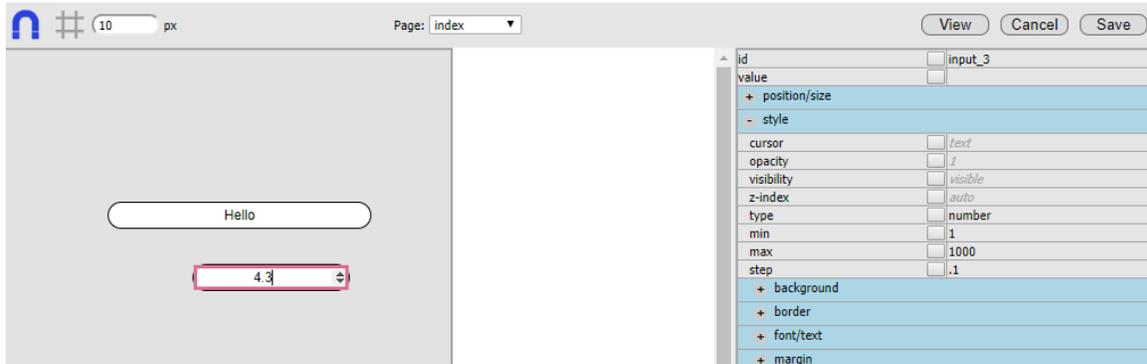
Html5 Panel Changes

Input box

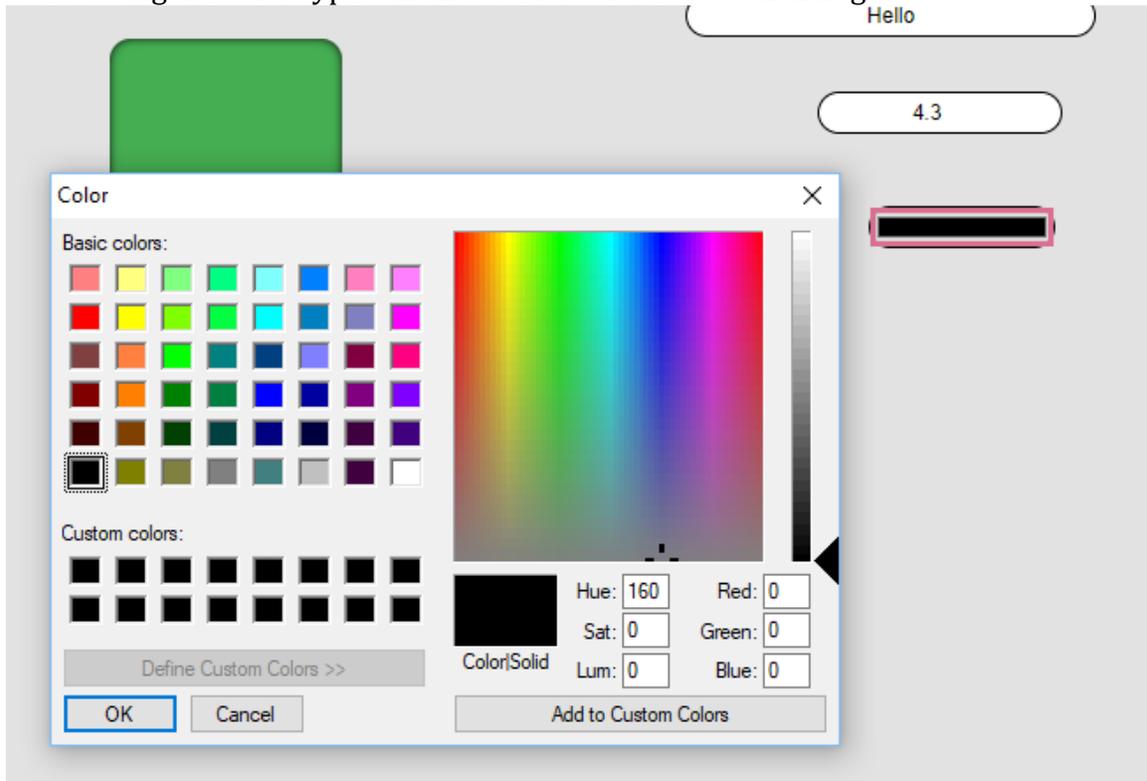
To see a video presentation on this feature visit:

http://pathfinderpc.com/pfcorepro_downloads/panelinputboxes.mp4

Input boxes have been added as an available element in Html5 panels. In the control selection there is now both a label and input box control. The input box allows user input. It includes a value property as well as a change and input event. The change event is raised when the user hits enter or moves off of the input field whereas the input event is raised as the user makes changes/types. Both of these events raise the current entered value. This allows for the capture of user input which can then be applied to logic flows to make decisions and changes. Additionally the input field has a type property which allows you to switch the input type between a number of different possibilities including text, numeric, color, etc. For example if you drag an input box onto a panel and change the type to be a number type and then fill in the min, max, and step properties you will obtain an input box with a set of arrows for incrementing and decrementing as well.



Or selecting the color type will allow for a color selection dialog.



Important Note: This is a beta feature. The types of input boxes that are available are obtained from the standard Html5 input element. Testing and additional work is still ongoing to make sure the correct parameters for each type are exposed.

To see a video presentation on this feature visit:

http://pathfinderpc.com/pfcorepro_downloads/panelinputboxes.mp4

Web browser (IFrame) Load Event

This version adds an event to the web browser control (iframe) which cycles false when the frame starts to load and true when the loading is complete.



Subpanels

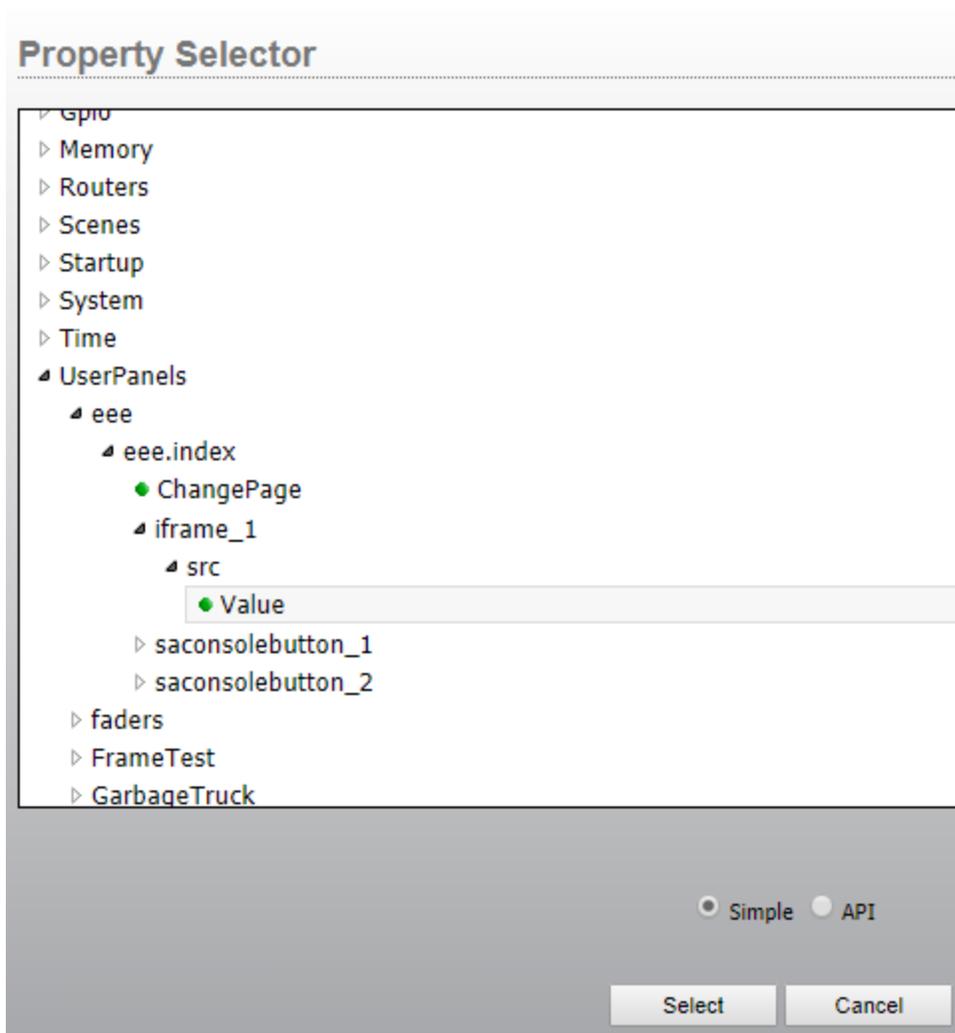
This is actually not a new feature but because several of the features below were created primarily to allow reusability of panel content and the associated messaging that goes along with that, it is worth discussing before we get into those topics.

PathfinderCore PRO Html 5 panels include a web browser component which is essentially just an Html5 iframe component. It can be used to embed other web pages and components (for example from a corporate page or video link) into a Pathfinder Core PRO panel. However it can also be used to embed a different panel into a parent panel. For example:



If you drag a web browser component into a new panel along with two buttons and then enable the binding on the src property of the web browser component, we can then use the button presses to load two different subpanels. Enable the binding for both mouse down and indicator on both buttons and save the panel.

Now in the mouse down property of button one, select the endpoint in the flow diagram and select the iframe src property as the endpoint.



For the translation make the mousedown true equal to:

`/userpanelframemin.php?panel=shared&page=page1`

Replacing the panel name (shared) and page name (page1) with the name of the panel and page you wish to display as a subpanel. Note that it is important to use the framemin.php page rather than the frame.php page so that the full PathfinderCore PRO header and menu system do not appear in the subpanel.

Translator Properties

True="/userpanelframemin.php?panel=shared&page=page1"

< Add Insert Before Insert After Remove >

True => /userpanelframemin.php?panel=shared&page=f

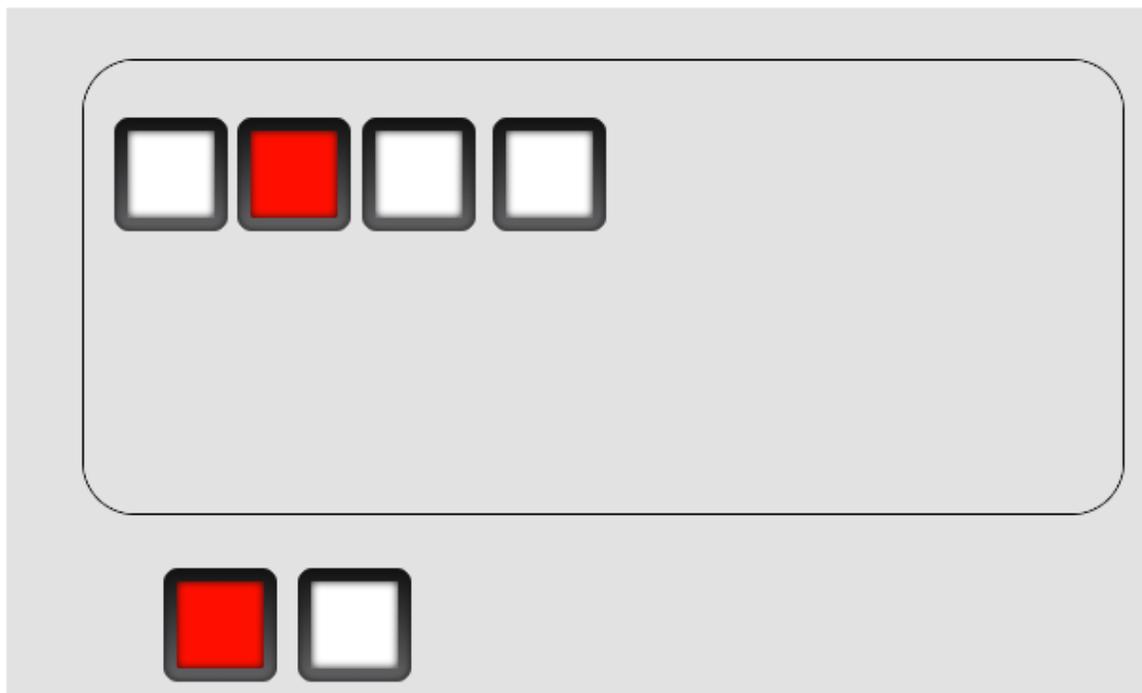
Advanced

Value Done Cancel Value

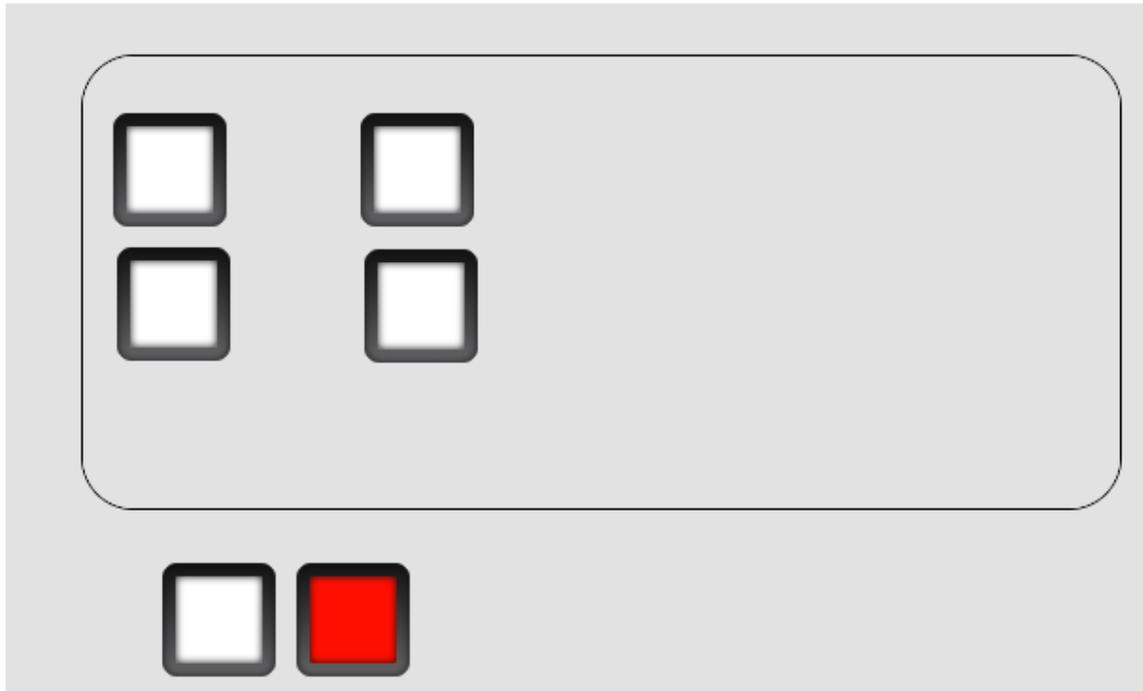
Click "Done" and then also allow the reverse binding to be set up so the indicator will light when the selected page is loaded in the iframe. Repeat the procedure for button 2 with a different page.

When the panel is executed, the web browser will now load the other panel as a subpanel of the main one. In this way we can create shared content that can be loaded in multiple panels as shown below:

Studio1



Studio1



Subpanel and shared content problems

The example above works great for many kinds of shared content. For example we could create a panel for each studio where one of the selection buttons loads a clock and meter subpanel and another loads an airchain display subpanel. And this works well with no other concerns.

The challenge comes when the shared subpanel needs to target different things depending on which parent it is loaded in. For example, what if the subpanel buttons in the examples above were source selectors but they needed to target a different destination depending whether they were loaded in Studio 1's parent panel or Studio 2's. The subpanel could be a specific and even paged set of selections that are common across all studios but would need to target the fader on which ever studio the subpanel was loaded in. The temptation to solve this problem would be to create some sort of logic flow that changes what happens depending on which panel last loaded the subpanel (Studio 1, 2, etc.) But that does not work because it does not allow for it to be loaded in both at the same time. And the subpanel itself has no knowledge that it is loaded as a subcomponent of another panel.

The only way to solve this problem is to introduce messaging that is internal to a given instance of a browser. We had to create a way for the subpanel to communicate with the parent panel instead of with a specific destination and vice

versa. That kind of messaging is the purpose of the Panel SetLocal and PanelMemorySlot features below.

Panel SetLocal

Note: This is considered an advanced feature and is often used in tandem with the Panel Memory slot below.

Each Html5 panel now includes a write only property called SetLocal. Selecting the overall panel will display this property in the property tree.



The purpose of this property is to send a message to set another property somewhere else on the running instance of the panel, its parent, or a subpanel. This property in addition to the Panel Memory Slots below is different in that it operates inside a given browser instance of a panel rather than upon all running instances of a given panel. In reality the distinction is a little more nuanced but will be explored more as we go through the shared subpanel example below.

In general this property will not have its value set in the property designer but rather the binding will be enabled and it can then be used/bound to other items in the panel via the panel's internal flows. The syntax that needs to be used for this property is:

```
<target>|<elementId>|<propertyName>=<PropertyValue>
```

So for example you could enable the binding of this property and then when a button is pressed you can set the value of this property to:

```
iframe_1|MyButton|Indicator=ON
```

The target portion can either be:

- parent = assumes this panel is loaded as a webbrowser (iframe) subpanel of another panel and we wish to send a property change to the parent panel.
- local = tries to set a property on the local instance of the panel
- name of a web browser (iframe) element in the form = tries to set a property on an element in a panel running in a web browser element in the existing panel. This variation looks for an iframe in the current panel of the correct name and then passes the property message to the web page running in that iframe.

Examples:

- parent|MyButton|Indicator=ON

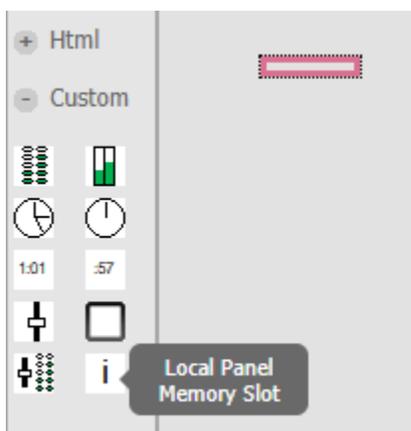
- Attempts to turn the Indicator property On for a Button named MyButton that is expected to reside on a panel in which this panel is running as a subpanel.
- local|MyButton|Indicator=ON
 - Attempts to turn the indicator property On for a button named My Button that is expected to reside on this instance of the panel.
- Iframe_1|MyButton|Indicator=ON
 - Attempts to turn the indicator property On for a button named MyButton that is expected to reside on a panel loaded in a web browser (Iframe) element called Iframe_1.

It is important to understand that setlocal sets properties in this instance of the browser only. If you have the studio 1 panel loaded on two different computers and you perform an action that uses the setlocal property to change an element's property, that change will only happen on that computer's instance of the panel unless other external bindings are also in use. In reality if the change is bound to a component inside the panel it will remain inside that instance of the panel, but if it is bound to something like a route change that is outside of the panel it will likely happen on all. This can be seen because using this property as a binding to a component inside the panel will not generate a panel based logic flow and binding it to something outside the panel will. The nuances of this are a bit subtle but should become clear as we work through the example below.

Panel Memory Slots

Note: This is considered an advanced feature and is often used in tandem with the Panel set local above.

Panel memory slots are similar to normal memory slots except that they only live inside a browser instance of a panel. They can be thought of as a javascript variable that also raises a change event inside the instance of the panel that is executing. The item can be found in the custom tool section. Dragging it onto a panel will create a dotted component. It is important to note that this component will be invisible when the panel is actually executing.



Similar to the setlocal property above, if this is bound using a panel flow to an item inside the panel (button, iframe, etc), no flow in the larger pathfinder will be created. Instead the value will just be changed or affect change within the running instance of the panel only. If the flow is bound to an item outside the panel (for example a route change of console fader state), then a flow will be created.

The Panel Memory slot has three important properties/events in the property grid:

- panelmemoryslotvalue = the value of the panel memory slot.
- raisetoserver = used to determine whether change events are raised only inside the local instance of the panel or also to the server.
- Panelmemoryslotchange = an event that fires when the value changes. This event carries the new value rather than just a true or false as its data.

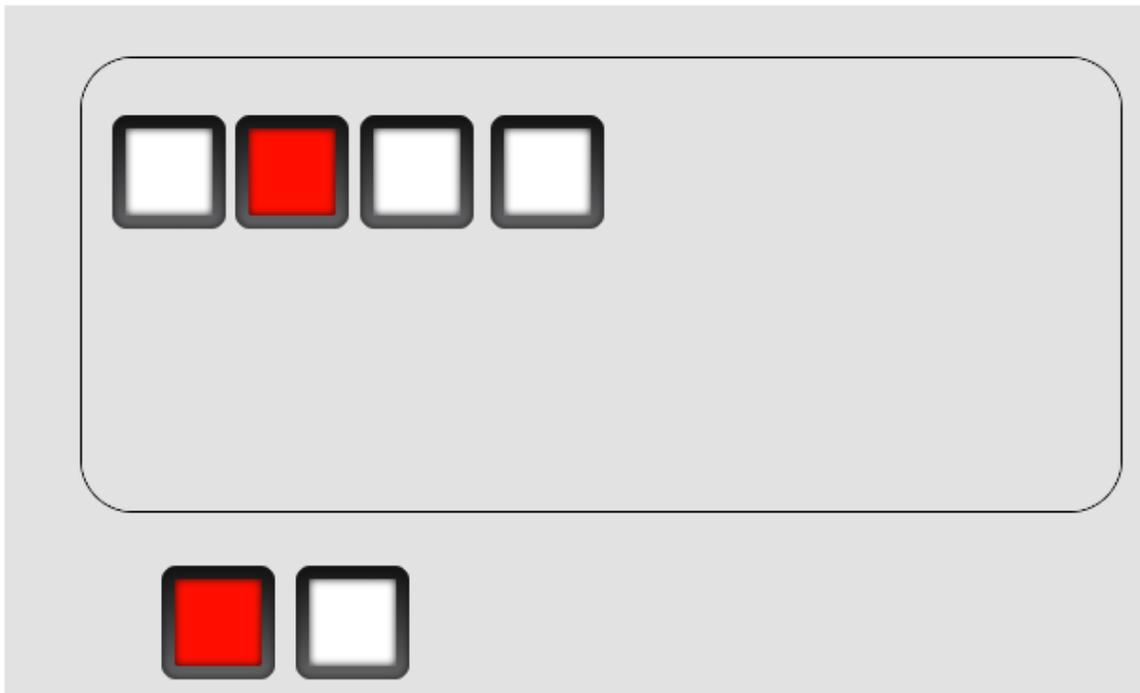
Shared Subpanel Example

Note: to see a video presentation of the functionality described below visit:

http://pathfinderpc.com/pfcorepro_downloads/reusable_subpanels.mp4

Returning to our example above:

Studio1



In this case what if the 4 buttons in the subpanel were source selectors? However, we want them to change the source on a destination that is dependent on which studio's parent panel the sub panel is loaded in. To accomplish this, we are going to use 3 panel memory slots. Two in the Studio 1 parent panel and 1 in the subpanel.

- Studio1 Panel
 - In the parent (Studio1) panel we will create two panel memory slots named currentsource and selectsource. We will enable the panelmemoryslotvalue and panelmemoryslotchange bindings for both slots and we will enable the binding on the SetLocal property of the panel and save.
 - currentsource
 - The panelmemoryslotvalue will be bound to the currentsource property of a virtual router destination using a panel flow and a *=* translation. This means that whenever the current source of that virtualrouter destination changes the number of the current source will be assigned to this studio 1 panel memoryslot.
 - The panelmemoryslotchange event will be bound to the Studio1 panel's setlocal property and the translation will look like: *=iframe_1|currentsource|panelmemoryslotvalue=*
 - The change event above means that whenever the currentsource value changes it will attempt to also set the same value on whatever panel is running in the subpanel loaded to iframe_1. Of course we have not created a currentsource memory slot on the panel that will be loaded as the subpanel yet but we will.
 - The entire purpose of this panel memory slot will be to pass the currentsource value to the instance of the subpanel loaded in iframe_1.
 - selectsource
 - The select source panelmemoryslotchange event will be tied to the same virtual router destination's current source property. Additionally on this slot the raisetoserver needs to be set to true. This slot will be used to change the route on the destination whenever its value changes.
- Shared Panel – page 1
 - In the Shared panel we will add a memory slot called currentsource to match and receive the current source from the parent's current source memory slot change event. We will enable the panelmemoryslotvalue and panelmemoryslotchange bindings on this panel memory slot. Additionally we need to enable the binding on the shared panel's setlocal property. And we need to save those changes.
 - We will bind each of the button indicators to the value of currentsource with a translation like:
 - 1=On
 - *=Off
 - Where 1 is the source whose button indicator we want to light

- When the panel is running, this will cause the subpanel to light the correct button depending on which source number has been fed to it from the parent.
- Last we will bind the button press to the Shared panel's setlocal property with a translation that looks like:
 - True=parent|selectsource|panelmemoryslotvalue=1
 - Where 1 is the source whose button has been pressed.
- This will allow the button press to pass the source selected to the parent panel's selectsource memory slot. Which if we remember is bound via its change event to the actual destination's currentsource.

The interesting part of the steps above is that we can now clone panel 1 to panel 2. In panel 2 we change the bindings on currentsource and selectsource to a different destination for Studio2. And we make sure that the panel selection buttons point to studio 2's iframe rather than studio 1's. But the shared panel does not need any changes. It will just start working for studio 2 and studio 2's destination. More importantly if we need to add functionality to the shared component it will work for both studios.

The key thought process that needs to be utilized is that if the shared content manipulates something specific then none of these steps are necessary. But if the shared content needs to manipulate something different depending on the parent panel it is loaded in, then we need to pass messaging between the internal panel and the parent to accomplish that.

This is one example of how these features may be used. It could be expanded greatly. For example the src property of the iframe could be assigned using a setlocal so that it does not need to be updated for each studio page. Or we could add additional panel memory slots for manipulating other destinations or even dynamically selecting the destination at the parent level.

Additionally there are future features planned for embedded routing components that may make this specific example less necessary. However, even if we add such components there will always be situations that are custom and do not work with our preconceived components.

It should also be pointed out that in many situations it is probably simpler, easier, and more understandable to just clone. Because this is a more advanced feature it is also more advanced to understand. However, for the times when the embedded content is duplicated in too many places and needs to be periodically changed, this allows for that kind of reuse.

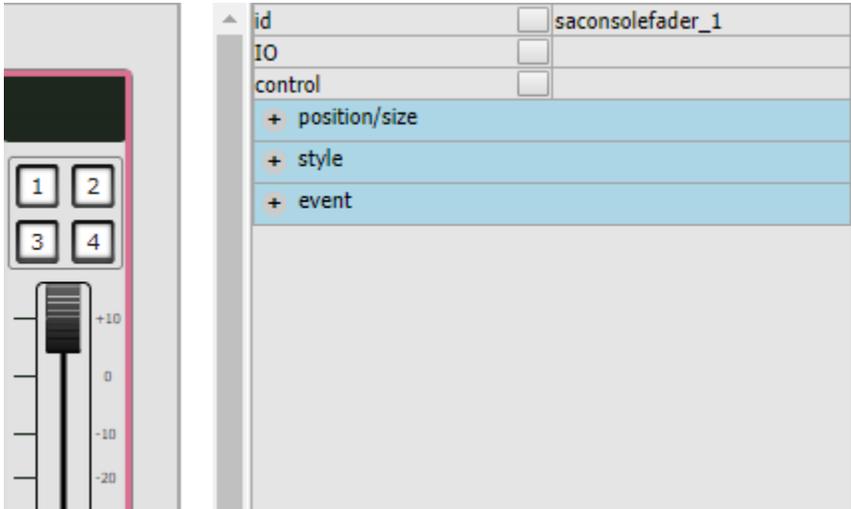
Note: to see a video presentation that may provide a clearer explanation of this functionality visit:

http://pathfinderpc.com/pfcorepro_downloads/reusablesubpanels.mp4

Version 1.5.9.08 Changes

Html5 Fader and Console Fader Control property (Lwch and Qor support)

This version adds the control property to Html5 panel Fader controls and Html5 panel Console Fader controls.



Traditionally a fader or a console fader is assigned its control point by using the IO selection which presents a list of sources and destinations in the main audio router. Selecting a specific source or destination will allow the fader to intelligently function and present the correct controls to the user of the panel. However, this functionality when applied to console faders assumes the use of Fach rather than Lwch.

Note: Fach is the control of the physical fader based on its position in the console where as Lwch allows control over a fader with a specific livewire channel loaded to it wherever it resides in the console.

Additionally some consoles (Qor based such as IQ and Radius) do not have a direct correlation between the Ios exposed to the router and faders those Ios may or may not be tied to. So this selection technique does not work in those cases.

The control property addresses this problem by adding an override of the control portion of the component. Clicking in the control property will present a list of control points that may be selected for the component.

Select Gain Control point

Device Name	IP Address	Type	Id
LWPS80236E	172.16.1.53	FaCH	8
LWPS80236E	172.16.1.53	FaCH	11
LWPS80236E	172.16.1.53	FaCH	7
LWPS80236E	172.16.1.53	FaCH	6
LWPS80236E	172.16.1.53	FaCH	2
LWPS80236E	172.16.1.53	FaCH	1
LWPS80236E	172.16.1.53	FaCH	13
LWPS80236E	172.16.1.53	FaCH	3
Element	172.16.1.51	LwCH	9501
Element	172.16.1.51	LwCH	21

Showing 21 to 30 of 48 entries

When the control property is used, the metering (in the case of console faders) is defined by the IO property, but the controls (fader, on/off, etc) are defined by the control property. If nothing is selected in the control property then the control point is inferred by the IO property. If nothing is selected in the IO property but the control property is used, then no metering will appear.

Using the control property, Lwch options and Qor faders are now available to these components where previously they were not available. It is important to note that Lwch and Qor faders do not have directly inferable metering points, so if metering is desired it must be manually selected with the IO property in addition to the control point with the control property. It is possible that in the future we may add code to attempt to figure out a correct metering point that would be correct for the assignment but for now it must be a manual decision. It is also important to note that the control property currently only displays fader objects as control point options. Xnode sources and destination control points and vmix points should be selected using the IO property. In the future those as well as XNode mix points may be selectable via the control property as well.

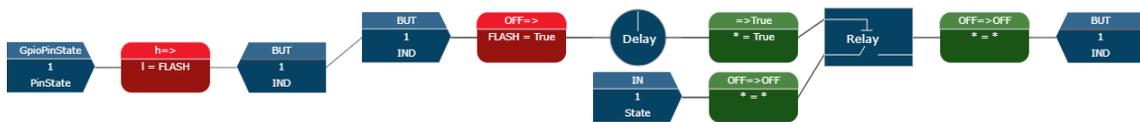
Version 1.5.11.14 Changes

PULSE in Html5 Button Indicators and LCD Button States

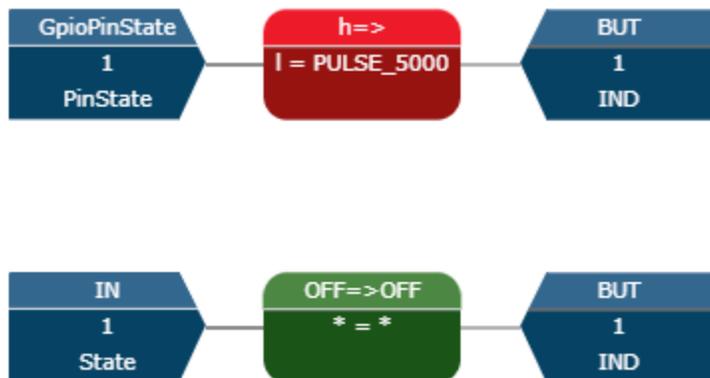
This version adds some additional values to html5 button indicators and LCD button states. The additional options are PULSE0 through PULSE10000. This feature will flash the button for the requisite number of milliseconds and then return to the last requested state. If the indicator state is changed while the pulse flash is in progress the new state will be returned once the flash time is complete. PULSE0 may be used to cancel an in progress pulse.

This allows the easy binding of state (ON/OFF) conditions while allowing an additional flow to control flashing.

For example prior to this feature the following logic might be required:



In this case a GPI causes the button to flash for 5 seconds. And after that the button either turns on or off depending on the state of VMIX 1. This type of flow can be useful for situations where you need to flash a button as an alert but retain a known state after the flashing is complete. With the addition of PULSE this becomes simpler.



In this case the button indication is tied to the VMIX state but setting pin 1 to low will cause a 5 second flash. It will then return to whatever state the vmix is currently in even if that state has changed during the duration of the pulse. This flow is much simpler.

If the button is an html5 button this can be made even simpler by binding the indicator to the vmix state within the user panel and only implementing the pulse (flash) flow in logic flows.

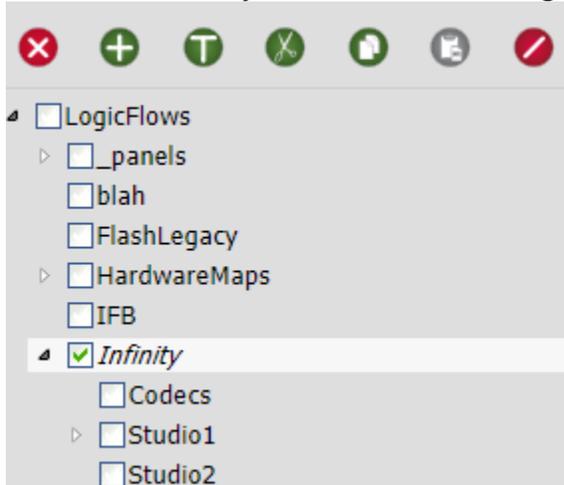
Version 1.5.12.15 Changes

Folder Disable Button

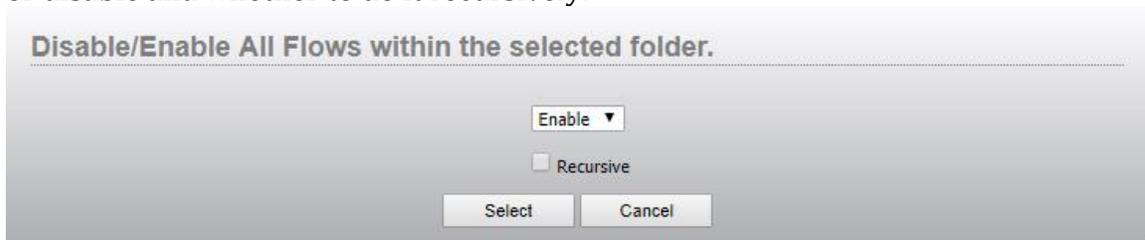
This version adds a disable button to the views in logic flows. It allows the logic flows designer to disable all the flows in a view and/or its sub views.



This button is only available when a single view is selected.



Clicking the enable/disable button will present a dialog to select whether to enable or disable and whether to do it recursively.



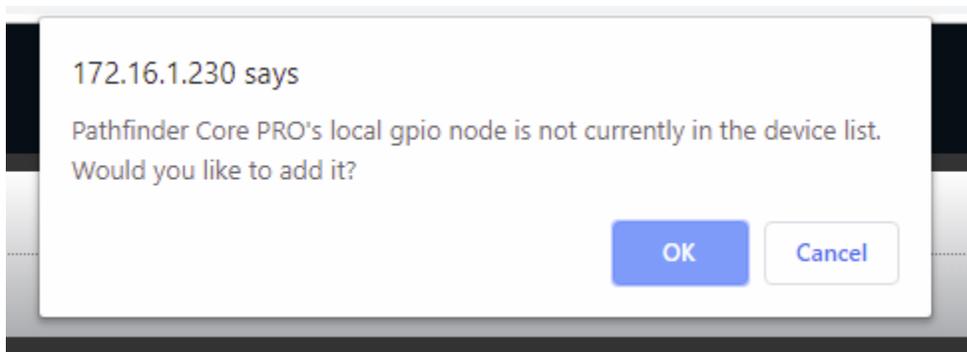
Selecting Enable or Disable and clicking the Select button will enable or disable all flows in the selected view. If the recursive option is used it will also enable or disable all flows in the sub views.

Version 1.5.16.18 Changes

Pathfinder Core PRO Gpio Node

Important Note: This functionality is a preview release. Cluster synchronization of this feature is not complete. See details below.

This version adds a virtual Gpio node inside Pathfinder Core PRO. This node may be accessed by browsing to the Gpio nav bar link under the System items. It behaves very much like the gpio portion of an Axia livewire driver except that the number of ports is dynamically adjustable and there are some additional advanced features to this node. This is also the avenue through which multicast gpio capability may be directly utilized by Pathfinder Core PRO. When you first access this web page it will detect that the internal gpio node has not been added to the devices table and will ask you if you wish to do so.



The node may only be used and/or configured after it has been added to the device list. Click OK to discover the internal Gpio node into the device list. This will also discover the gpio points into the gpio router as well.

Pathfinder Core PRO Gpio 

Port Count:

Show 10 entries

id	Name	MulticastMode	IRoutedTo	ORoutedTo	SourceAddress	Edit
1	GPIO_1	NODE	0	NONE	9501	edit
2	GPIO_2	NODE	0	NONE		edit
3	GPIO_3	NODE	0	NONE		edit
4	GPIO_4	NODE	0	NONE		edit

Showing 1 to 4 of 4 entries

Once the device has been added you will see the list of gpio ports. By default, there are 4 ports in the system. You can add additional gpio ports by changing the port count at the top of the screen. This allows you to add any number of gpio ports into the local virtual gpio node.

Use the edit link to change the property values for any Gpio port.

Edit Gpio Port

Port Id: 1

Port Name:

MCast Mode:

IRoute Mode:

ORoute Mode:

Source Address:

GPIO Refresher

It is important to understand the various ways Gpios may be used in an Axia environment.

- Software Gpios with no source address assignment will simply allow closures to be directly tripped on either GPI or GPO by Pathfinder Core PRO.
 - Example: SRCA:""
- If a source address field for a Gpio port uses a livewire channel number in its address field, this means it will listen to and generate closures to and from an Axia Console over multicast.
 - Example: SRCA:9501
- Finally, if the source address uses an IPAddress/port format it means that the gpio port should use TCP to connect to the device at that ip address and monitor the GPIs from the selected port on that device and mirror those closures on this port's Gpos. This is the method used by the PathfinderCore PRO gpio router to route gpio data across a network.
 - Example: SRCA:"172.16.1.23/8"

Pathfinder Core PRO Gpio Properties

Pathfinder Core PRO's gpio node works in the same way as described above by default. However, it adds a few additional properties for manipulation by advanced users. The use of each of the Pathfinder Core PRO properties is described below. These advanced properties are available by using the edit link on each port.

Edit Gpio Port

Port Id: 1

Port Name:

MCast Mode:

IRoute Mode:

ORoute Mode:

Source Address:

- Name: This allows you to assign a unique name to each gpio port.
- Source Address: This field may be left blank if the closure will be used by Pathfinder and does not need to follow another port or console's functionality. Or you can assign either a livewire channel number or ip/port value to this field. Note that for ip/port, you do not need to assign it in the configuration user interface. The gpio router will allow you to manipulate this field much more easily by just making route changes.
- Multicast Mode: By default, this will be set to Node.
 - Node: In this mode using a livewire channel number in the source address will cause the port to behave in the same manner as a livewire driver gpio port. Specifically, the port will listen to gpio messages from the console fader on which the livewire channel number is loaded and change its GPO pins accordingly. And tripping the GPI pins of the port will send closures to the Axia Console.
 - Console: Changing the mode to console means the port will behave as if it were an Axia console. Therefore, GPIs that are tripped on this node will be sent to the GPOs of other Gpio ports on the network that have the same channel number assigned. And tripping a GPI on one of those other devices will cause the GPO on this port to change.
- IRoutedTo: By default, this will be set to 0. This property is only applicable to situations where ip/port or livewire channel numbering is used in the source address. By default, when a GPI comes in from either of these sources it is applied to a GPO on the port. This property allows you to change that and apply it to the GPI instead or to None which means it is thrown out.
- ORoutedTo: Similar to IRouted To, this property defines where inbound GPOs will be sent. By default, they are sent nowhere as this setting replicates other gpio node functionality. When using IP/port source addressing the Pathfinder Core PRO gpio node is unique in that it will also subscribe to GPO changes for the port and these may be routed to I, O, or None.

Using IRouteTo and ORoutedTo it is possible to make a multicast or unicast gpio route where one ports I's and O's mirror another port.

Important Clustering Notes: Version 1.5.16.18 is a preview build of the Pathfinder Core PRO internal gpio node functionality. Cluster synchronization is not yet complete. For example, if you use this node in a cluster the route changes and pin state changes will not be synchronized between the nodes of the cluster. Additionally, ip/port route changes where the Pathfinder Core PRO internal node is the source and other equipment is the destination will not work properly on the remote device if the secondary node in the cluster takes over. This is because the external IP address of the first node was used in the route change. These issues will be addressed in a future build, but we thought it was important to make this available for single node users and for systems integrators to start providing feedback while we work through the more complex cluster synchronization issues.

Version 1.5.16.19 Changes

Pathfinder Core PRO Gpio Node Clustering

Important Note: Pcp Gpio Node is still very new as it was only introduced in 1.5.16.18. This version adds clustering synchronization to the feature. Please report any issues you encounter with the functionality.

This version adds synchronization of the PathfinderCore PRO Gpio Node in a cluster. Port additions and removals as well as port configurations applied to either PathfinderCore PRO of the cluster will be replicated to the other PathfinderCore PRO in the cluster. Pin closure states will also be replicated.

There are certain situations where cluster replication of pin states will not occur. Specifically, if the port has been configured with a multicast gpio channel or snake routing assignment (IP/port), replication becomes a bit more nuanced. Closures that are directly tripped will still be replicated. However, closures that are sensed from the mcast or unicast snake (GPIs from the other snake port) will not be replicated. This is because it is assumed that both cluster nodes are monitoring those changes directly from the third piece of equipment and will pick up their states directly from the other end of the snake. Therefore, cluster replication of that state would in fact be redundant and could lead to race conditions.

When dealing with unicast snake mode (IP/port assignments), there are some additional clustering nuances to be aware of. These are most easily discussed via an example. When you route a gpio node to the PathfinderCore PRO gpio port, the address field in the gpio port on both cluster nodes will get filled by ipaddress/port. For example:

- PathfinderCore PRO A = 172.16.1.241
- PathfinderCore PRO B = 172.16.1.242
- Gpio XNode = 172.16.1.85

If you route (using the PathfinderCore PRO Gpio Router) the XNode gpio source 1 to the PathfinderCore PRO A or B destination 1, the destination port on both PathfinderCore PROs will look like:

- CFG GPO 1 SRCA:"172.16.1.85/1"

In this example both PathfinderCore PROs are monitoring the GPI changes on port 1 of XNode at 172.16.1.85 and making matching changes on their gpo pins. This is straight forward.

However, if we route either of the PathfinderCore PRO servers to the XNode Gpio things get a bit more interesting. We can only apply one IP address to the SRCA field

on the XNode destination. Therefore, Pathfinder Core PRO will use the Axia livewire address of whichever PathfinderCore PRO in the cluster currently has its event system active. In the example above Pathfinder Core PRO A would typically be the active server in the cluster. When you make the route change on either PathfinderCore PRO, the system will use the active server IP address on the XNode destination. The XNode destination would look like:

- CFG GPO 1 SRCA:"172.16.1.241/1"

If Pathfinder Core PRO A Server were to fail or get shut down, the B Server will loop through its routes and find any destination that PathfinderCore PRO A's gpio ports were routed to and switch them to Pathfinder Core PRO B's external livewire ip address.

- A Fails
- XNode port gets reassigned:
 - CFG GPO 1 SRCA:"172.16.1.242/1"

If A comes back online the port will switch back to use the A server when the event system comes online again.

One important note about this is that the XNode gpio clears its pin states to be all high each time a route change happens and then reassigns them to the new values based on the new GPIs. This means that if a snake route is held low on the XNode based on a low GPI on PathfinderCore PRO, during a cluster failover those pins might flicker to high and then back to low. This is usually a virtually instantaneous flip. This should only cause an issue if you have a PathfinderCore PRO GPI that is unicast snake routed to an XNode (or other Axia gpio device) GPO and that closure is both normally held low and you have an activity (such as an automation advance) that happens on the flip from high to low and you fail over to the secondary node in the cluster. But this is also similar to what would happen if a device supplying that same closure were to be restarted. Note that this is not an issue for multicast snake routing.

In the future we may address this issue by using floating IPs where the ip route would not change but rather the ip address used for the external routes would float/move between the Pathfinder Core PRO devices.

We highly recommend that users do some testing of this functionality to fully understand how it works. For the vast majority of use cases it should be seamless.

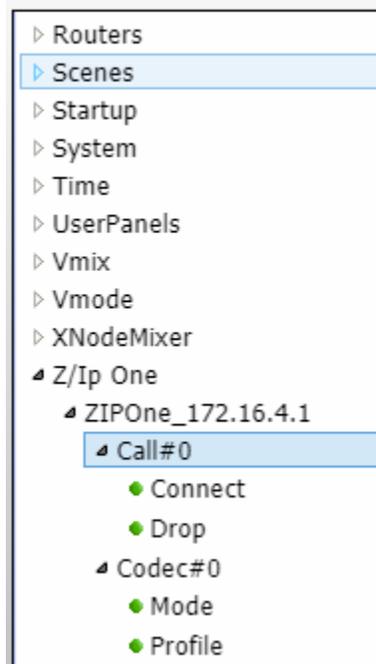
Version 1.5.18.21 Changes

Pathfinder Core PRO Z/IP One Control

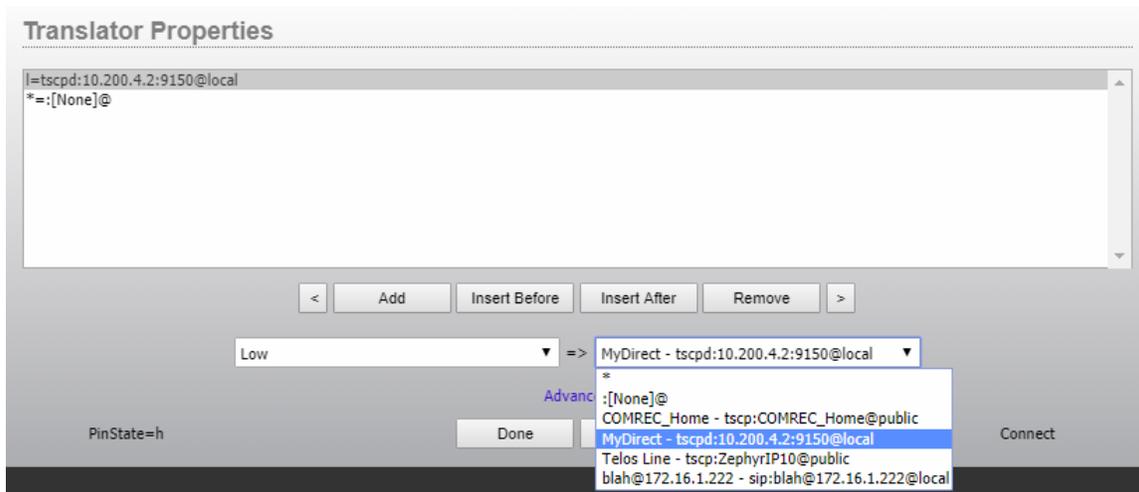
This version adds many additional control points for the Z/IP one which are available via Logic Flows. Dialing and status items are exposed in the simple tree in logic flows under a new Z/IP One branch and many more properties are exposed in the API tree under the device itself.

When selecting an End Point in the Simple tree there will now be a Z/IP One branch with several properties for any Zip One in the system.

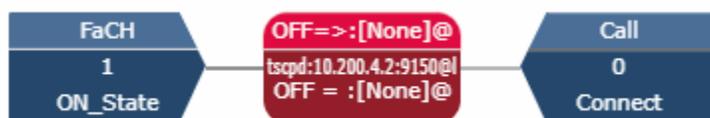
Property Selector



Connect and drop can be used to establish and drop connections. In general, the Codec settings will not be used as an endpoint since those settings are normally stored within the call phone book entries. After selecting the Connect property as a logic flow end point, the translation list will display the list of entries in the Zip One phone book. Only phone book entries may be used for dialing currently.



There is also a special phonebook entry that only lives in PathfinderCore PRO called None which when dialed will behave the same as the Drop item. For example:



This flow will dial the Zip one every time Fader 1 is turned on and drop the call every time it is turned off.

When viewing a start point there are some different options.

- Z/Ip One
 - ZIPOne_172.16.4.1
 - Call#0
 - State
 - Codec#0
 - Mode
 - Profile
 - ConnectedTo
 - ConnectionState
 - Online

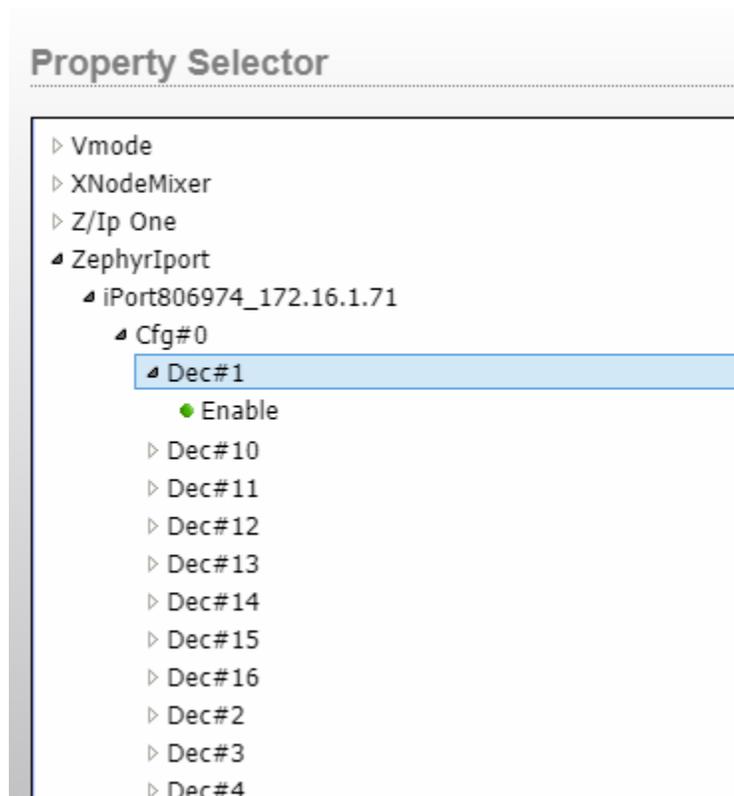
It is important to understand the difference between Call#0 state and ConnectionState. Call state will list a myriad of possible connection statuses including multiple options for idle and how the connection was established. ConnectionState reduces this down to much simpler Idle, Connecting, and Connected options. As such it is recommended for most situations where logic needs to be fired based on the connection being active or disconnected. The ConnectedTo property will contain the matching phone book entry to the one

selected in the Call Connect property example above. Therefore, you can display the current call connection phonebook entry name.

In addition to these properties, the API tree contains a much larger set of properties matching from the Z/Ip One Lwcp API. Consult the Zip One documentation for details on many of these other properties.

Pathfinder Core PRO IPort Control

This version adds many additional control points for the IPort. In the simple tree there will be a new IPort branch which will expose the Encoder and Decoder enable and disable options. We have limited those options only to the simple tree as most of the other properties related to the Zip One mpeg side of the configuration are considered advanced and not normally dynamically adjusted.



Most of the other settings found on the Configuration pages of the Iport are available via the API. Because the API matches the object layout of the device's Lwcp protocol some of the new settings can be found in a variety of places. The new paths include:

LwcpInterpreter#0.LwcpRoot#0.Decoder#
LwcpInterpreter#0.LwcpRoot#0.Encoder#

LwrpInterpreter#0.LwrpRoot#0.Cfg#0.Enc#
LwrpInterpreter#0.LwrpRoot#0.Cfg#0.Dec#

For example, the destination properties for a decoder or encoder are available via the LwrpRoot#0.Cfg#0.Enc and LwrpRoot#0.Cfg#0.Dec paths whereas the encoder algorithm and bitrate settings are available via the LwrpRoot#0.Encoder and LwrpRoot#0.Decoder paths.

Additionally, there is a LwrpRoot#0.Cfg#0.Gpio path that controls whether the gpio ports are enabled or disabled in the system. The actual Gpio port may or may not be present depending on whether the encoder or decoder are active. All of these paths are of course available for use in logic flows via the API tree. Please note that some are read only and so will only appear when using a start point.

Tip: To figure out which properties match config options in the Iport UI, open a TCP connection to PathfinderCore PRO using Putty to port 9600 and subscribe to changes on the device in question.

```
sub Devices#0.ZephyrIPort#[tcp://172.16.1.71:93] $MAX_DEPTH=-1
```

Then you can make changes in the IPort's UI and see the changed properties in Core PRO. This can be useful to see where certain properties are located in the API tree.

Version 1.5.19.23 Changes

The changes in this version described below all apply to the Html5 user panels.

Digital and Analog Countdown Timer Changes

Both the digital and analog countdown timers now have some additional properties.

- **Elapsed Event:** This is an event that can be raised when the count down timer completes.
- **StopMode:** This property defines what happens when the countdownstart is set to false while the countdown is progressing. The options are:
 - **StopAndReset:** stopping the timer by setting countdownstart to false will cause the timer to stop and be reset to the countdown value.
 - **Pause:** stopping the timer by setting countdownstart to false will cause the timer to stop where it is in the countdown process and hold that value. Setting countdownstart to true again will cause the countdown to continue from where it left off.
- **CountUp:** Changing this value to True instead of False will cause the timer to count up rather than down to the selected countdownlength. If countdownlength is zero it will count up indefinitely until stopped and/or reset.
- **Reset:** This is an action property that can be used by logic flows and/or bindings to reset the counter. Setting it to true will cause the timer to reset.
- **ResetMode:** This property defines what happens when a reset is issued and a countdown is in progress. The options are:
 - **ResetAndStop:** Setting reset to true will cause the timer to reset and stop its countdown.
 - **ResetAndContinue:** If the timer is running this will cause it to reset and continue running. If the timer is stopped this will just reset the value.

It is important to note that currently the timer does not automatically reset when the countdown completes. We may add an option for that in the future depending on customer feedback. If that functionality is desired, it can be obtained by using the property bindings to hook the elapsed event to the reset state such that when the timer elapses reset is triggered. This can be accomplished either by using the flow in elapsed to change the reset or by using the flow in reset to change based on the elapsed start point. In both cases the panel must be saved with the binding buttons for these properties/events turned on to find those options in the flow property tree.

For example:

id	<input type="checkbox"/>	sadigitalcountdown_1
countdownlength	<input checked="" type="checkbox"/>	10
countdownstart	<input checked="" type="checkbox"/>	false
stopmode	<input checked="" type="checkbox"/>	Pause
countup	<input type="checkbox"/>	false
reset	<input checked="" type="checkbox"/>	false
resetmode	<input checked="" type="checkbox"/>	ResetAndStop
+ position/size		
+ style		
- event		
mousedown	<input type="checkbox"/>	
mouseup	<input type="checkbox"/>	
elapsed	<input checked="" type="checkbox"/>	

In the picture above the binding buttons for both elapsed and reset have been enabled and the panel has been saved with those buttons enabled. Now select the reset property field and click the start point in the bottom corner. Then browser into the user panels, the specific panel, and the digital countdown control, and select the Elapsed property:

Property Selector

▼ UserPanel#QorLange

- ▲ UserPanel#testbed
 - ▲ testbed.index
 - ▶ AnalogReset
 - ▶ AnalogStart
 - ▶ AnalogStop
 - ChangePage
 - ▶ DigitalReset
 - ▶ DigitalStart
 - ▶ DigitalStop
 - ▶ saanalogcountdown_1
 - ▶ saconsoleknob_1
 - ▲ sadigitalcountdown_1
 - ▶ --countdownlength
 - ▶ --countdownstart
 - ▲ --elapsed

● Value
 - ▶ --reset
 - ▶ --resetmode

Simple API

Select True=True for the translation

Translator Properties

True=True

< Add Insert Before Insert After Remove >

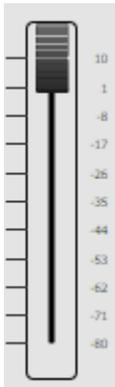
True => True

Advanced

Value Done Cancel

And click Done. After saving the panel if the countdown elapses the timer should automatically get reset.

Fader Control Changes



- The Fader control will now display the value of the fader if you hover over the fader.
- The mouse wheel can now be used to manipulate the fader.

Controlling non-fader numeric properties

The fader can now be tied to values other than just traditional console faders. Using the control property there will be an additional button in the dialog to switch to the typical logic flow endpoint tree:

Select Gain Control point

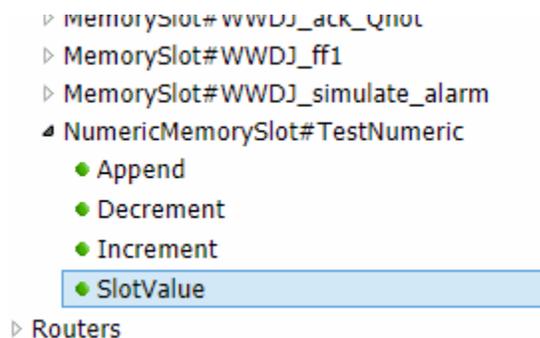
Show entries

Device Name	IP Address
Element	172.16.1.51
LWPS80236E	172.16.1.53

Showing 1 to 10 of 40 entries

Select Cancel None Endpoints

Clicking the Endpoints button will bring up that tree. From there you can pick any numeric property in the tree to connect with. If you select a non-numeric property the dialog will give you a warning. You should avoid using non-numeric properties with the control property. See below how to use non-numeric properties. For example, we could select a numeric memory slot as the control point for the fader:



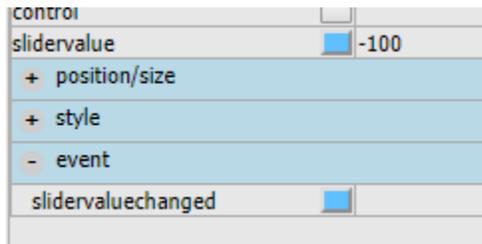
After saving the panel the fader would now control and update based on the value of the memory slot. In order for this to work properly there are several other properties that can be used:

- Min: the minimum value allowable by the fader
- Max: the maximum value allowable by the fader
- Step: the steps that can be used for the change. A value of 1 would mean the value has to be integers whereas a step value of .1 would allow for a single decimal place in the values.
- Type: This property has two options:
 - Audio: the scale of the fader will follow a typical audio fader where larger moves in the optimal range of the fader relate to smaller decibel changes.
 - Linear: the scale is a direct linear scale.

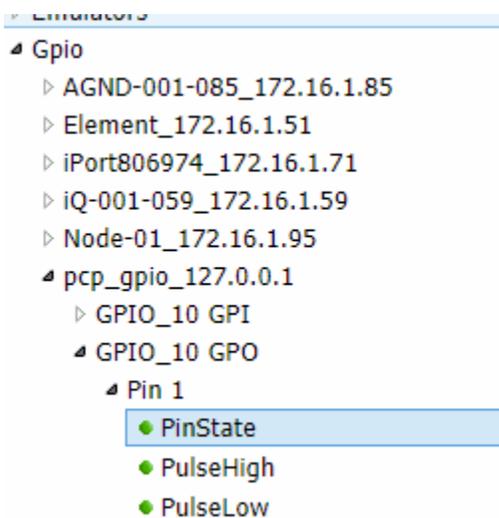
It is important to note that when using the control property to select the functionality for the fader, the system will try to automatically set the min, max, and step properties if the system knows what they should be for the given control point.

Controlling non-numeric properties

Another option is to not use the control property to bind the functionality and instead to bind the display value and value change properties directly to properties. This can be useful in situations where a translation is necessary. For example, we could connect a fader up to a true/false or gpio value. In this case we do not use the control property but instead we would enable the binding on the slidervalue property and the slidervaluechanged event.



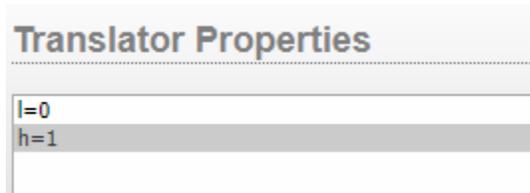
Set the min and max for 0 and 1 and the step for 1 and the type to linear. Next set the slidervaluechanged event to the gpio pinstate of a gpio port.



For the translation translate 0 to low and 1 to high.



Then in the SliderValue property select the same GPO pinstate as the endpoint and reverse the translation. That will make sure that if something else changes the gpio pin that it will be updated in the fader:



For this example you may also want to turn the metrics off since there are only two valid states.

This is probably not the best use case for this functionality, but it shows how you can use the fader control to manipulate any value in the system either directly using the control property if it is a numeric property or via translation and the SliderValue and SliderValueChanged property and event.

Console Knob

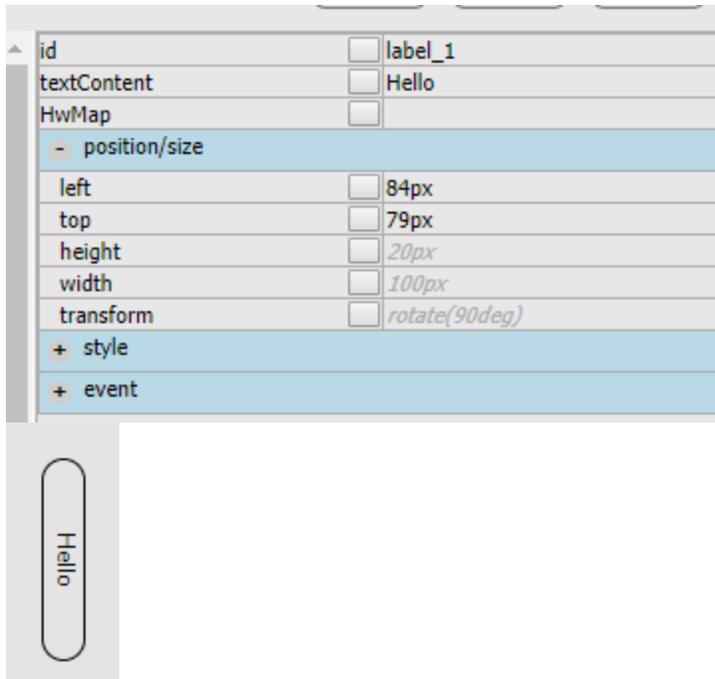
This version adds a rotary knob that may be used in the same way as the fader above. If you have not reviewed the fader changes above, please do so as the same properties for numeric and non-numeric control as well as min, max, step, etc. properties also apply to the console knob object.



This control can also have its control property used to select a console fader or other numeric property in the system to control in the same manner as described for the fader control above. In addition, this component also shares the indicator property with button objects so that the knob color may be changed based on some indication state.

Transform Property

This version exposes the transform css property for use in html components. This can be used to rotate, skew, and scale objects. For example, you can rotate a label by creating the label and then entering rotate(90deg) into the transform css style.

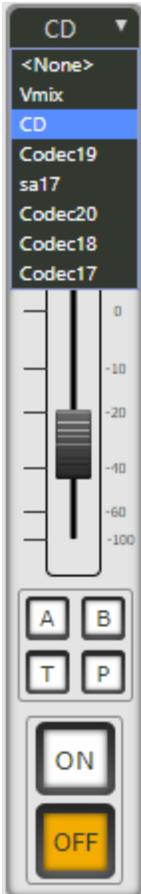


It is important to note that not all components (especially the more complex ones) will transform well and work properly when rotated. Also when you return to the object in the designer it may not show the simple value you entered as the html will translate that into a matrix to accomplish the transform. For more details about the transform property see:

https://www.w3schools.com/cssref/css3_pr_transform.asp

Console Fader Control Changes

The Console Fader now includes a drop down list for source profile selection when it is connected to a console fader. For example:



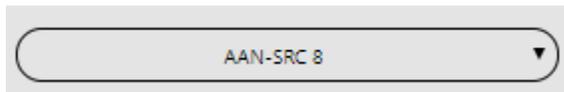
The list is obtained from the enabled source profiles for the fader in question. This feature may also be disabled if you do not want your panel users to be able to change the source profile on a fader. This can be accomplished using the `allowsourceprofilechange` property of the console fader.

id	<input type="checkbox"/>	saconsolefader_1
IO	<input type="checkbox"/>	tcp://172.16.1.53:93?l=DST&d=...
allowsourceprofilechange	<input type="checkbox"/>	true
control	<input type="checkbox"/>	

Shift Edit Property List

This version adds an option for advanced users where holding the shift key while clicking in the user panel property editor fields will bypass the usual helper dialogs and allow direct editing of the text. This can be useful for things like copying and pasting color values. Shift click to highlight the text box without the helper dialog and then click again to edit the text editing for the property.

List Selector



This component is still a work in progress. Significant user interface work needs to be done to make the usage of this component intuitive to configure. Until that work is done this component is available for advanced users that understand the API. This component provides a drop-down list for selecting elements in the system. For example, it could be used as a source selector for a virtual router or a show profile selector for a console. Unfortunately, until we finish a more intuitive configuration user interface, some knowledge of the API and inner working of PathfinderCore PRO are required to configure this component. A thorough understanding of SapV2 (Appendix A) will help in the configuration of this component. Also feel free to reach out to support for guidance.

There are 4 properties that are used together to describe the list options and another two for event and state:

- **Listsearchpath:** This holds a SapV2 object path and optional property value for the root from which all objects in the list will be obtained. For example:
 - `Routers#0.VirtualRouter#4 SapObjectType=VirtualSource`
 - This specifies that we will be filling the list with data from virtual sources in virtual router number 4.
 - `Devices#0.Qor#[tcp://172.16.1.59:93].LwcpInterpreter#0.LwcpRoot#0.AppControl#0 ObjectName=ShowProfile`
 - This specified that we will be filling the list with data from show profiles on the Qor at 172.16.1.59.
- **Listsearchdepth:** The number of branches below the listsearch path root to look for elements. Many times, this will be 1 for one level deeper than the root selection, but it could be higher or -1 for infinite. For Example:
 - `Devices#0.Qor#[tcp://172.16.1.59:93].LwcpInterpreter#0.LwcpRoot#0.AppControl#0.ShowProfile#0`
 - This is one branch deeper than the search path in the last example above.
- **ItemDisplayProperty:** This is the property that will be used as the display data for each object. This is what the user sees. In both of the examples above this would likely be the name property.
- **Itemselectvalue:** This is the property whose value will be used in the change and current value events. In the examples above it would be the Id and/or ObjectId properties.
- **CurrentValue:** this can be used by logic flows to select the displayed value.
- **Change event:** this can be used to make a change when a different item in the drop down is selected. The value that will be available in the translation is defined by the itemdisplayproperty.

Let's work through two examples.

List Selector Example 1

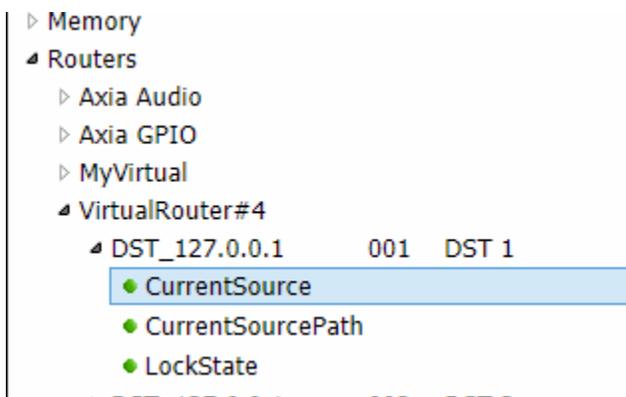
In example 1 we will use the selector to present a list of sources from a virtual router. When the user selects a different source it will change a specific destination on that router. Make sure you have a virtual router in the system. In this case we will use VirtualRouter 4. Set the following property values:

- Listsearchpath: Routers#0.VirtualRouter#4 SapOb
- Listsearchdepth: 1
- Itemdisplayproperty: Name
- Itemdisplayvalue: Id

id	<input type="checkbox"/>	saselector_1
HwMap	<input type="checkbox"/>	
listsearchpath	<input type="checkbox"/>	Routers#0.VirtualRouter#4 SapOb
listsearchdepth	<input type="checkbox"/>	1
itemdisplayproperty	<input type="checkbox"/>	Name
itemselectproperty	<input type="checkbox"/>	Id

This means that we are looking for objects up to 1 level deep below the Routers#0.VirtualRouter#4 path whose SapObjectType=VirtualSource. The last part of this is important so that the selector does not also show destinations. For each source that is found the selector will create an item in the select list whose display value is taken from the name property and whose select value is taken from the id property. This will display a list of sources by name from virtual router 4 and the value used when a selection is made will be the source's id.

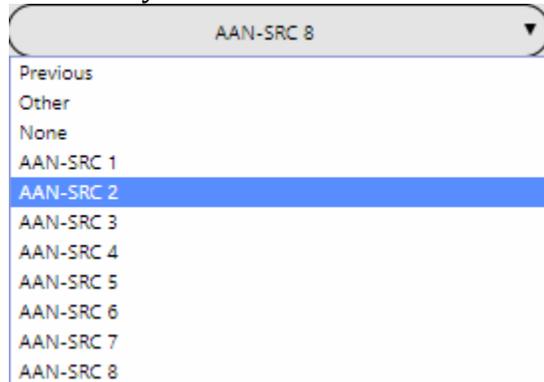
Next enable the binding on the currentvalue and change properties. For the change property select Destination 1's current source property in the virtual router.



And use *=* for the translation. This will make sure that any time a new selection is made in the list, the number of that source will be passed to the destination's currentsource property thereby affecting a route change.

For the currentvalue property, select destination 1's current source again as the start point. Translation would again be *=*. This will ensure that if the destination route changes even by some other means that the drop down list will change to show the correct source.

In this way we have created a route selector from a drop down list.



List Selector Example 2

In example 2 we will use the selector to present a list of show profiles for a qor/iq. When the user selects a different show profile for the qor/iq, it will change the show profile. Set the following property values (modifying for your own device):

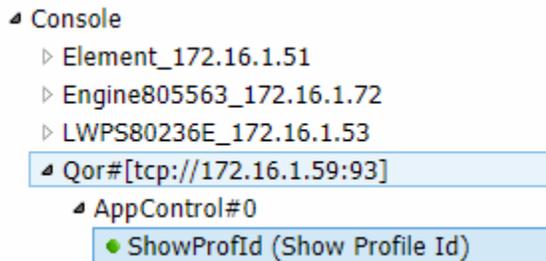
- Listsearchpath:
Devices#0.Qor#[tcp://172.16.1.59:93].LwcpInterpreter#0.LwcpRoot#0.AppControl#0 ObjectName=ShowProfile
- Listsearchdepth: 1
- Itemdisplayproperty: Name
- Itemdisplayvalue: ObjectId

id	<input type="checkbox"/>	saselector_2
HwMap	<input type="checkbox"/>	
listsearchpath	<input type="checkbox"/>	Devices#0.Qor#[tcp://172.16.1.59
listsearchdepth	<input type="checkbox"/>	1
itemdisplayproperty	<input type="checkbox"/>	Name
itemselectproperty	<input type="checkbox"/>	ObjectId

This means that we are looking for objects up to 1 level deep below the Devices#0.Qor#[tcp://172.16.1.59:93].LwcpInterpreter#0.LwcpRoot#0.AppControl#0 path whose SapObjectType=ShowProfile. For each show profile that is found the selector will create an item in the select list whose display value is taken from the name property and whose select value is taken from the ObjectId property. This

will display a list of sources by name from Qor/Iq in question and the value used when a selection is made will be the show profile's id.

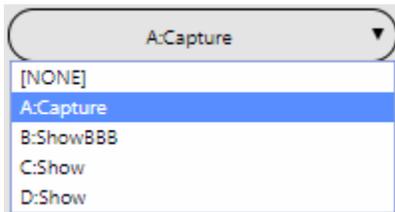
Next enable the binding on the currentvalue and change properties. For the change property select the console's appcontrol ShowProfId property.



And use `*=*` for the translation. This will make sure that any time a new selection is made in the list, the number of that show profile will be passed to the console's showprofid property thereby affecting a show profile change.

For the currentvalue property select showprofid property again as the start point. Again use `*=*` for the translation. This will ensure that if the show profile changes even by some other means, that the drop-down list will change to show the correct show profile.

In this way we have created a drop-down list for selecting a new show profile.



We realize that until more intuitive user configuration user interfaces are created, this component may be challenging. If you need to understand how to use it for a specific task, please reach out to support and realize that this is a beta feature and may require some time for them to obtain the correct parameters for your task.

Version 1.5.19.24 Changes

Timestamp Memory Slot

This version adds a time stamp memory slot that can be used to grab time stamps. This can be useful in displaying the last time something happened in a user panel.

Memory Slot Editor

Type:	<input type="text" value="Time Stamp Memory Slot"/>
Name:	<input type="text" value="MyStamp"/>
Value:	<input type="text"/>
Startup State:	<input type="text" value="Blank"/>
Pattern:	<input type="text"/>
Persistent:	<input type="text" value="True"/>

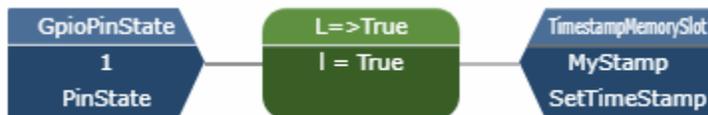
When you first create a time stamp memory slot, you will probably not apply a value. The pattern field can also be left blank. It can be used to specify the format of the time stamp. We will describe that in more detail below. After creating the timestamp memory slot, it can be used in logic flows by applying an endpoint to the SetTimeStamp write only property of the memory slot.

Property Selector

Property Selector dialog box showing the selection of the `SetTimeStamp` property under the `TimestampMemorySlot#MyStamp` folder. The dialog includes the following text and controls:

- Set current time into the memory slot
- Simple API
- Select Cancel

For example:



This logic flow will set the memory slot to the current date and time every time the specified gpio pin goes low.

Memory Slots

Show entries

Name	
MyStamp	2019-06-13T05:58:52.524-04:00

The format of the date time stamp can be changed using the pattern field. If you edit the memory slot again you will see that the default format has been assigned to the memory slot:

Pattern:

yyyy-MM-ddTHH:mm:ss.fffzzz

The possibly patterns follow the .net pattern for date/time strings as specified at:

<https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings>

and

<https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings>

For example:

Using a pattern of just D will return:

Thursday, 13 June 2019

Using a pattern of s will return:

2019-06-13T06:41:20

More examples (examples in us – see links for other languages):

d => 6/15/2009

D => Monday, June 15, 2009

f => Monday, June 15, 2009 1:45 PM

F => Monday, June 15, 2009 1:45:30 PM

yyyy-MM-ddTHH:mm:ss => 2019-06-13T06:41:20

Version 1.5.19.25 Changes

Web UI Code changes

One of the challenges with working in a web-based environment is the speed at which both the languages and tools change. It is not uncommon to attend a conference where the gurus are avowing a certain technology for molding javascript as the thing we should all be using just to hear the same guru 2 years later saying – “yeah; I was wrong about that – this is what we should be doing”. This can make it challenging to keep the code viable for original and new team members on a long-term project. And periodically it makes sense to reevaluate.

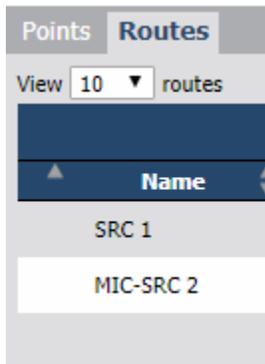
This build is the product of a month’s worth of code refactoring in order to begin making use of some of the more current javascript development tools including nodejs, webpack, and babel. The code has been broken up into much more easily manageable files and then packed back together using web pack during the build process for deployment. This allows us to move forward with the code in a more efficient manner while maintaining and improving efficiency with the browser as well. In order to accomplish this almost all the web page code files have been tweaked, modified, and manipulated in some fashion over the course of several hundred code repository commits. Whenever as much code changes as has in this build there is the possibility of bugs. While we have attempted to test thoroughly, if anything is not working in your environment please be ready to report the issue to support so that we can fix it as quickly as possible.

Data Table changes

In the course of refactoring the code mentioned above we also updated to a new version of the library we use to present web-based data tables. Unfortunately, between the original version and the version we are now using the tool underwent its own overhaul of its API. In order to make use of the new features this meant modifying and/or rewriting much of the code that uses and populates those data tables. However, this also allowed us to make use of a new feature called deferred rendering. In previous versions and table rows were rendered in the browser whether or not they were visible. In the new version this data is kept in background datasets and only rendered as html as needed. This is much more efficient and can reduce the load time of very large datasets significantly. It also paves the way for some more changes we have planned to improve the performance with large in browser data sets even more. Again, please contact support if you encounter any issues when using this version so we may address them as soon as possible.

Additional sort field in routes table

This version adds an additional sort icon on the left edge of the routes table.



The screenshot shows a web interface with two tabs: 'Points' and 'Routes'. The 'Routes' tab is active. Below the tabs is a 'View' dropdown menu set to '10 routes'. Below the menu is a table with a dark blue header row containing the text 'Name'. On the left side of this header row, there is a small white triangle pointing upwards, which is the sort icon. Below the header, the table contains two rows: 'SRC 1' and 'MIC-SRC 2'.

Name
SRC 1
MIC-SRC 2

This sort icon will sort according to the destination's ip address and io number where those are organized numerically rather than by text.

Version 1.5.19.26 Changes

Qor/Iq LwcpSs support

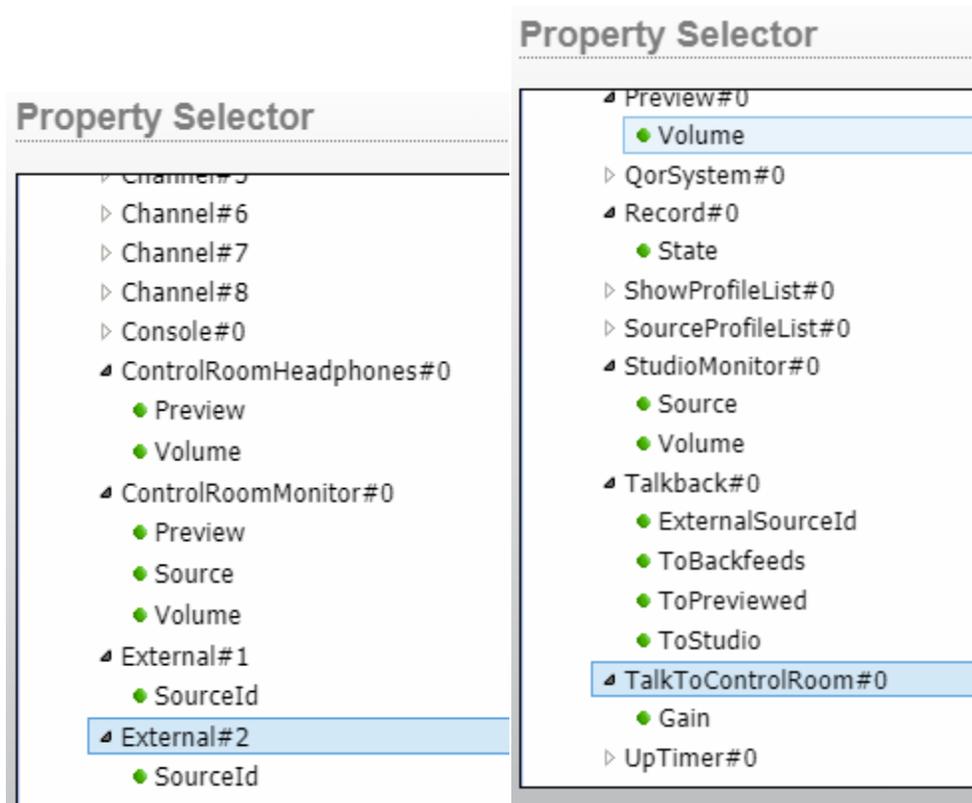
This version adds support for an additional port and protocol exposed in current versions of Qor and Iqx. That port offers additional control points via logic flows including monitor section control as well as eq and dynamics. These will be available under the Console control branch for the Iq/Qor in question. For example:

The image displays two screenshots of a 'Property Selector' interface. The left screenshot shows a tree view with the following structure:

- iQ-001-059_172.16.1.59
- ▾ iQx-Demo_172.16.1.58
- AppControl#0
- ▾ Channel#1
 - ChannelMute
- ▾ Dynamics#0
 - Compressor#0
 - DeEsser#0
 - NoiseGate#0
- ▾ Equalizer#0
 - Active
 - FaderGain
 - OnSwitch
 - PanoramaPosition
 - Preview
 - Program1
 - Program2
 - Program3

The right screenshot shows a list of properties for 'Talk#0':

- Active
- ▾ Equalizer#0
 - Active
- FaderGain
- OnSwitch
- PanoramaPosition
- Preview
- Program1
- Program2
- Program3
- Program4
- ▾ Source#0
 - Id
- ▾ Talk#0
 - ToControlRoomSwitch
 - ToPreviewedSwitch
 - ToStudioSwitch
- Talkback



Clicking on any of these new properties will provide a description as to what the property is used for. Additionally, there are even more parameters that are exposed by this protocol under the device in the API tree. Note that this protocol only exists in Qor and iQx based consoles. Also, the protocol within the console itself is still considered experimental. While all tests so far are working well, please report any issues that do not work as expected so that we may investigate.

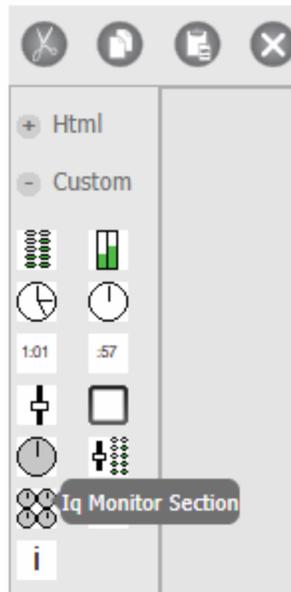
Because this is very much a beta feature, the support for this protocol may also be disabled through the advanced options (Configuration, edit advanced options) by adding the following parameter:

```
SET Devices#0 LwcpSs=False
```

Qor/Iq Monitor section html5 user panel component

This version also adds an additional component to user panels that acts as a monitor section for an Iq/Qor based console.

[New Panel]



Dragging this component onto a panel will present a monitor section for an Iq/Qor console:



In order to assign it to a specific Iqx or Qor based console, select the component in the panel and click the control property:

id	<input type="checkbox"/>	saiqmonitor_1
control	<input type="checkbox"/>	

In the list of available control points you will find one for each Iq/Qor whose type is LwcpSsRoot. Those are the only control points that may be used with this component.

Select Gain Control point

Show 10 entries Search Filenames:

Device Name	IP Address	Type	Id
iQx-Demo	172.16.1.58	LwcpSsRoot	0
iQ-001-059	172.16.1.59	FaCH	1
iQ-001-059	172.16.1.59	FaCH	2
iQ-001-059	172.16.1.59	FaCH	3
iQ-001-059	172.16.1.59	FaCH	4
iQ-001-059	172.16.1.59	FaCH	5
iQ-001-059	172.16.1.59	FaCH	6
iQ-001-059	172.16.1.59	FaCH	7
iQ-001-059	172.16.1.59	FaCH	8
iQ-001-059	172.16.1.59	LwcpSsRoot	0

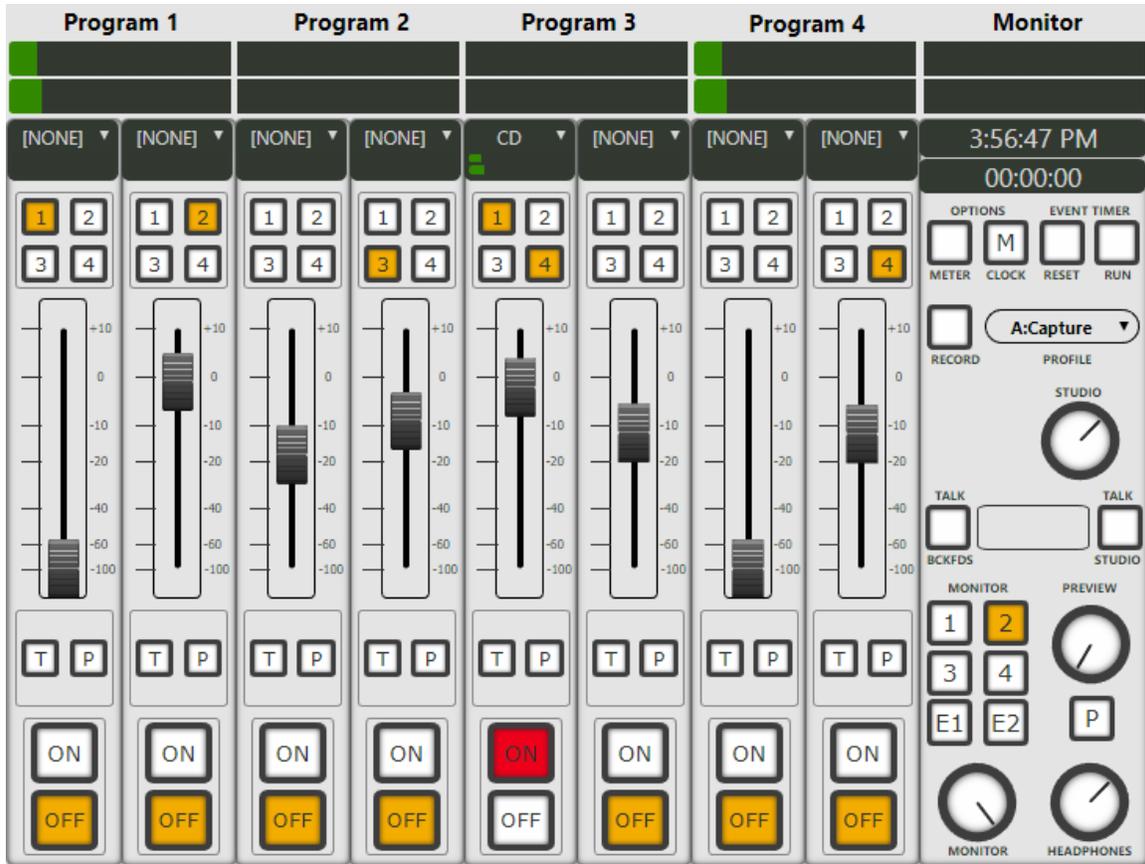
Showing 41 to 50 of 50 entries 1 row selected

First Previous 1 2 3 4 5 Next Last

Select Cancel None

In the future we will probably limit the list to only these options when the monitor section component is selected. Also note that the LwcpSs support discussed above must be enabled in order to use this component.

Using the console fader controls and this monitor section it is pretty easy to create a console control panel in PathfinderCore PRO.



There are a few points that need to be made about this monitor section component:

- The meter option button currently does not do anything. It is for future use.
- The clock and timer functionality is controlled by PathfinderCore PRO and not the console and so will not be in sync with the console timer and countdown clock. This is because those control points are not currently available in the LwcpSs control protocol. That may change at some point in the future.
- As this is also very much a beta feature it may also be disabled using an advanced option via the edit options button on the configuration page:
 - SET Devices#0 QorMonitor=False

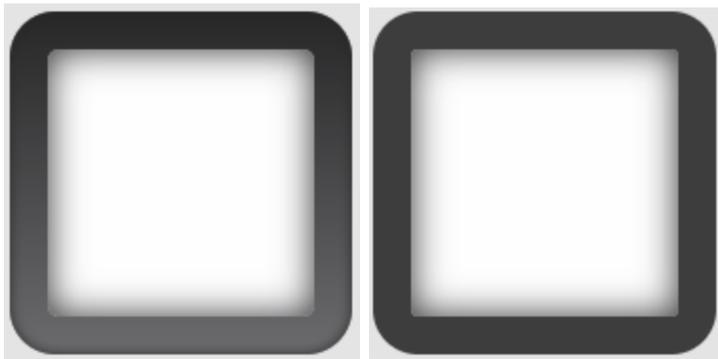
Please contact support if you need assistance or encounter any issues with this new feature.

Html5 User Panel Console Button border-gradient property

This version adds a new property for console buttons called border-gradient.

border		
border-color	<input type="checkbox"/>	#000000
border-width	<input type="checkbox"/>	0px
border-radius	<input type="checkbox"/>	13.3%
border-style	<input type="checkbox"/>	none
border-collapse	<input type="checkbox"/>	separate
box-shadow	<input type="checkbox"/>	rgb(31, 31, 32) 0px 0px 6px 0p...
border-gradient	<input type="checkbox"/>	complex

No value or a value of complex will yield the same button style as before whereas a value of simple will remove the background top to bottom gradient.

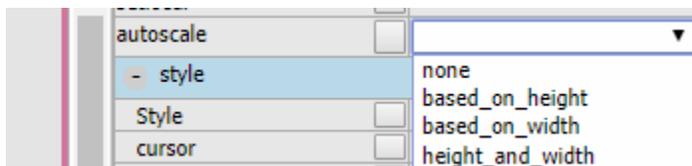


The reason we have made this additional property has to do with sizing. The original design is a button centered in a background. But as you resize, this can cause the button to be off by a pixel one way or the other. At smaller sizes this can become an objectionable artifact. The simple design is just a button with a border and therefore the border scales more perfectly. The first has a slightly softer feel while the second scales better.

It is important to note that this version uses the simple mode for the console fader which will alter the look of your existing panels that use that component slightly.

Html5 User Panel Autoscaling

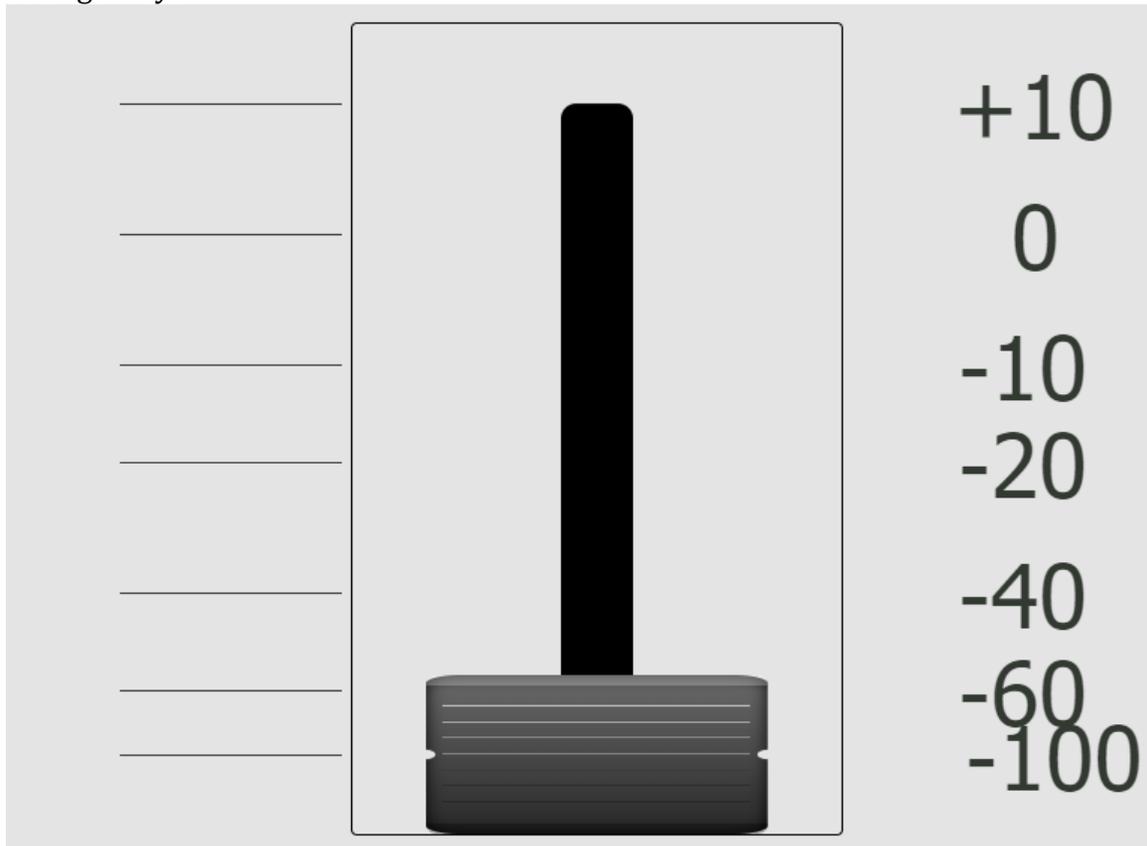
This version adds an additional property to panels themselves called autoscale. To see it select the user panel background in the designer.



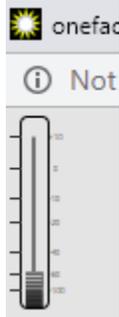
No value or a value of none will work the way panels have always worked. The other options will automatically change the size of the panel components according

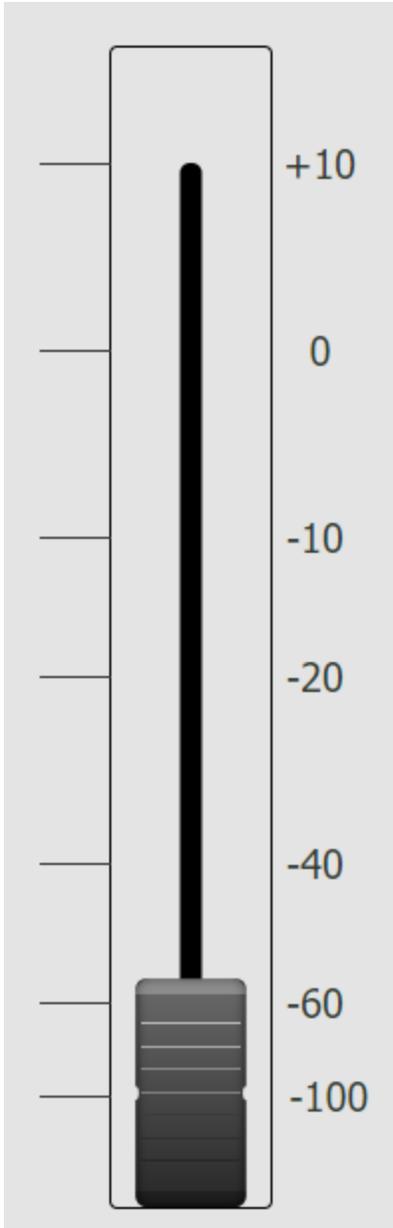
to a change in the browser window dimensions. If you select `based_on_height`, changes in the browser's height will cause the amount of height change to be used as the factor for scaling up height and width of the components. Changing the browser width will do nothing. Conversely selecting `based_on_width` will cause the amount of width change to be used as the factor for scaling up height and width of the components. Changing the browser height would do nothing. Setting it to `height_and_width` will cause the height change to be used for the height difference of the components and the width change to be used for the width difference.

While intuitively you might believe that `height_and_width` is the correct choice, it rarely is. That is because it causes buttons and components to be stretched in strange ways.



On the other hand scaling based on height alone will maintain the correct aspect of the component while just scaling it up and down:





In general, if the aspect ratio of the panel objects is important and the screen it will be displayed on is a widescreen monitor it is best to choose scaling based on height. If it is a wide screen monitor turned for portrait display then the best option is based on width. If aspect ratio does not matter like in the case of pure buttons where square or rectangle does not matter then based on height and width may be used.

It is best to create the panel in question and save it with one of the scale options and then play with the browser size to get a feel for which option is most appropriate. You need to refresh the panel after you resave with a change to the option to see the effect.

Also note that there appears to be a bug in this version with the `height_and_width` option in relation to the console fader and monitor section components. Use `based_on_height` or `based_on_width` for panels with these components until a patch is created.

Version 1.5.20.28 Changes

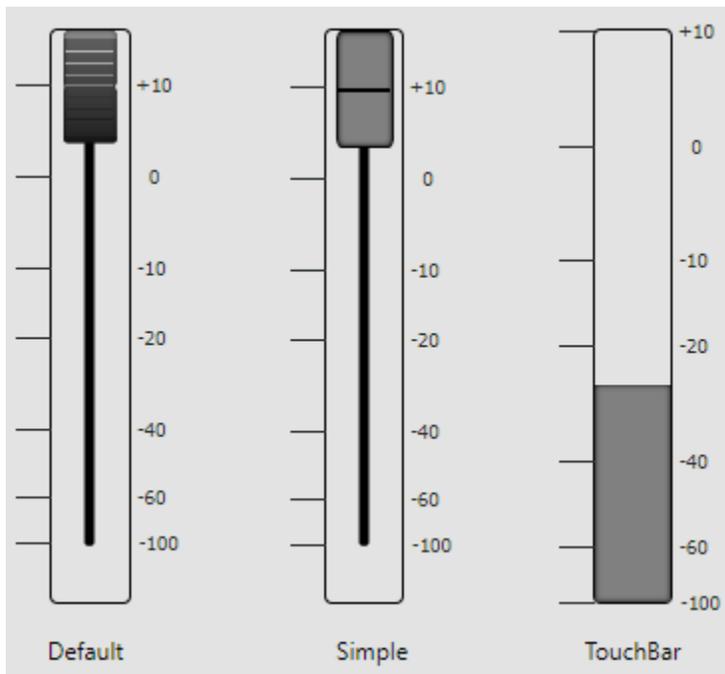
Important note: The changes below required a number of changes to the user panel default skin css file. If the components discussed below do not appear correctly in the panel designer web page, your executing panel, or in the router-details web page, try forcing the browser to refresh its cache. In Chrome this can be accomplished by holding Ctrl and Shift while clicking the reload this page icon  while on the web page having difficulties. You should only have to do this once on each of these three pages and then the browser will fetch and cache the revised stylesheet file.

Fader user panel component

This version adds several properties to the Fader user panel component. Most of these properties are exposed under the style section header in the property list for the component.

faderstyle property

Fader objects can now have one of three different styles: default, simple, and touchbar.



Default represents the same fader style as PathfinderCore PRO has had previously. The simple style removes the complexity of the fader and turns it into a simple

rectangle with rounded edges and an optional center line. The touchbar style is designed for touch interfaces. In this variation dragging with your finger (using a touch screen) or mouse anywhere in the fader rectangle will cause the bar level to go up or down. This is particularly suited to touch screens (especially smaller ones) as it increases the area that is available for touch manipulation rather than requiring a grab of the smaller specific fader object.

orientation property

Fader objects can now be oriented either horizontally or vertically (the default) using the orientation property.

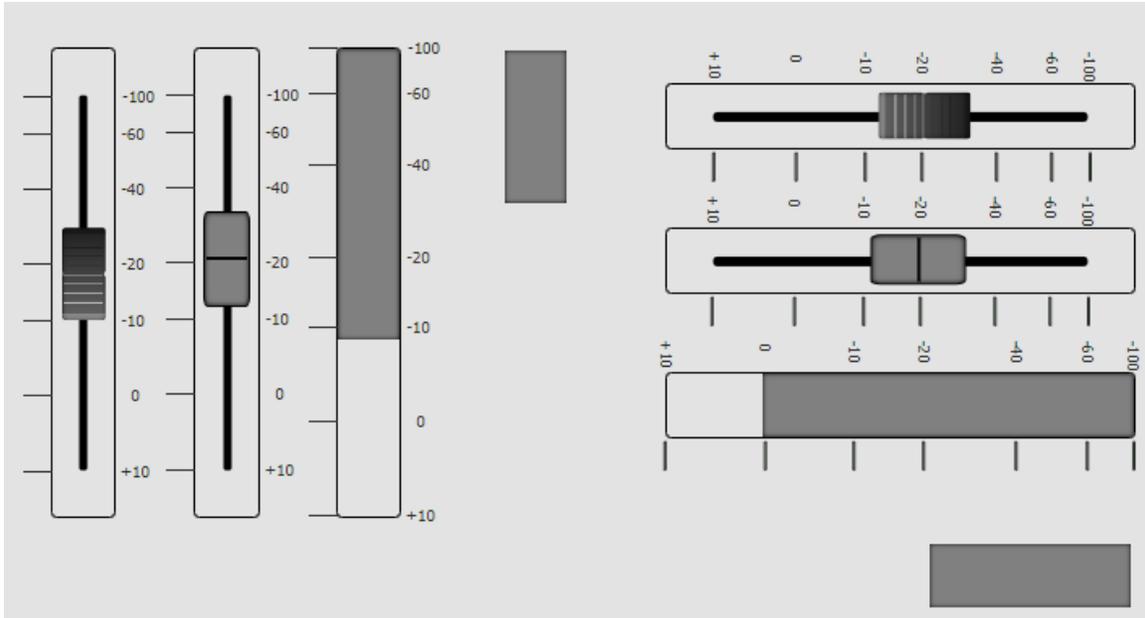


It is important to note that there is still some work to be done here with this property. By default, when you flip the orientation, the height and width are retained rather than reversed as expected making for a squashed funky graphic. This will be fixed in a future build. However, in the interim you can hold the shift key while resizing to stretch it into the correct size. Sometimes a save is necessary to get the slider width to settle in properly. Another solution is to manually swap the width and height values in the property list and then clicking save in order to set the initial correct height to width scaling ratio. These issues will be resolved in a future build.

Invert properties

Faders also now include the properties: invert, motioninvert, and metrictextinvert. The invert property may be used to flip the up/down or left/right functioning of the fader. The motion invert and metrictextinvert should be used along side the invert property in order to set the mouse, scroll wheel, and metric layouts to match the functional inversion. For example, it may be desired for the wheel up functionality to still increase the gain even though the fader is now upside down or it may be

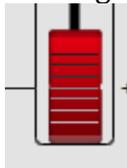
desirable to have the wheel motion follow the fader up and down state on the screen. Since faders may also be used to control other numeric values instead of just audio this opens up more of those possibilities. As always, it is also possible to turn the metrics off and use the faders that way as well.



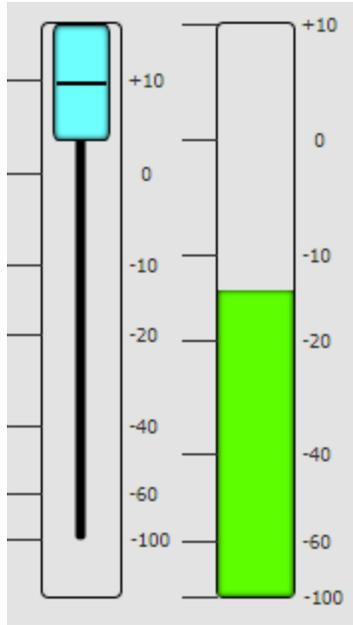
Color and style properties

Under the style/slider property sections are a number of new properties for styling the actual slider. These include:

- **slider-width:** can be used to alter the width of the slider.
- **slider-opacity:** can be used to alter the opacity of the slider.
- **slider-background:** Primarily used by the default fader style, this can produce a background gradient. For example, to turn the default fader red, one might change the background to a value like:
 - `linear-gradient(#770000, #ff0000)`



- **slider-back-color:** Primarily used for the simple and touch bar styles, this will set the background color of the simple slider or of the touchbar.



-
- Default fader style properties
 - **slider-top-color**: adjusts the color of the top edge of the default fader style. Due to opacity blending, this can be a subtle change.
 - **slider-bottom-color**: adjusts the color of the bottom edge of the default fader style. Due to opacity blending, this can be a subtle change.
 - **slider-default-line0-color**: adjusts the color of the first line on the default fader.
 - **slider-default-line1-color**: adjusts the color of the second line on the default fader.
 - **slider-default-line2-color**: adjusts the color of the third line on the default fader.
 - **slider-default-line3-color**: adjusts the color of the fourth line on the default fader.
 - **slider-default-line4-color**: adjusts the color of the fifth line on the default fader.
 - **slider-default-line5-color**: adjusts the color of the sixth line on the default fader.
 - **slider-default-line6-color**: adjusts the color of the seventh line on the default fader.
- Simple fader style properties
 - **slider-simple-line-color**: adjusts the color of the line on the simple fader style.
 - **slider-simple-line-display**: adjusts whether the line on the simple fader style is displayed or not.

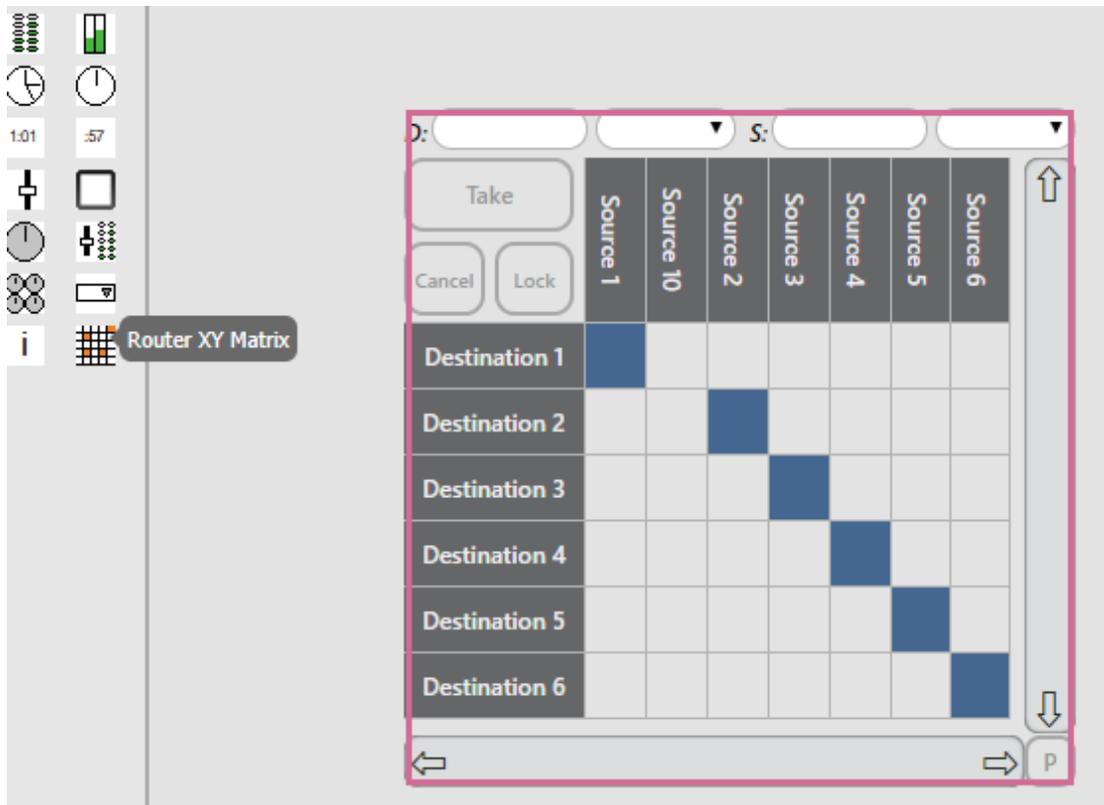
Console Fader user panel component

Many of the properties discussed above in the fader object have also been exposed in the Console Fader so that the color and style of the slider within the larger console fader object may be manipulated. These styles correspond directly to the styles above and include:

- fader-slider-background
- fader-slider-back-color
- fader-slider-top-color
- fader-slider-bottom-color
- fader-slider-default-line0-color
- fader-slider-default-line1-color
- fader-slider-default-line2-color
- fader-slider-default-line3-color
- fader-slider-default-line4-color
- fader-slider-default-line5-color
- fader-slider-default-line6-color
- fader-slider-simple-line-color
- fader-slider-simple-line-display
- fader-faderstyle

Router XY Matrix user panel component

This version adds a Router XY Matrix component both as a user panel component and as an additional tab on the router-details page. It can be added to a panel using the RouterXY Matrix component.



After adding the component to the page, resize it and then using the router property select a router to use with it.

The screenshot shows an XY Matrix interface with a grid of 10 sources and 10 destinations. The sources are labeled Source 1 through Source 10, and the destinations are labeled Destination 1 through Destination 10. The grid contains blue squares indicating connections between sources and destinations. The connections are as follows:

Destination \ Source	Source 1	Source 2	Source 3	Source 4	Source 5	Source 6	Source 7	Source 8	Source 9	Source 10
Destination 1	Connected									
Destination 2		Connected								
Destination 3			Connected							
Destination 4				Connected						
Destination 5					Connected					
Destination 6						Connected				
Destination 7							Connected			
Destination 8								Connected		
Destination 9									Connected	
Destination 10		Connected								



Select Router

Show 10 entries Search Filenames:

Id	Name	Description	Type
1	Axia Audio	Axia Audio Router	AxiaAudio
2	Axia GPIO	Axia Gpio Router	AxiaGPIO
3	MyVirtual		Virtual
4	NewVirt		Virtual

Showing 1 to 4 of 4 entries

First Previous 1 Next Last

Select Cancel None

It is important to note that the component will not fill with actual router data while viewed in the designer. This means that the actual column and row header size may be inflated since the default designer example population only has 10 sources and destinations. It is useful to view it in the actual executing panel to see how it will function.

Using the XY Matrix

Once a matrix has been added to a panel, a router assigned to it, and the panel saved. View the panel to see how it functions.

The screenshot shows a routing table interface. At the top, there are two search bars labeled 'D:' and 'S:', both containing the text 'None'. Below these are two dropdown menus, also labeled 'D:' and 'S:', both set to 'None'. On the left side, there are three buttons: 'Take', 'Cancel', and 'Lock'. The main area is a grid with columns representing sources and rows representing destinations. The columns are: Program 1, Program 2, Program 4 R..., Aux Send 1, Aux Send 2, CR Monitor ..., CR Monitor, CR Headpho..., Preview, Talk to CR, Guest Head..., Studio Monitor ON LWPS80236E, Talent Hdph..., Talkback, Program 3, Program 4, Aux Send 3, and Aux Send 4. The rows are: Channel 1, Channel 10, Return 1, Return 2, CR Monitor, CR Headpho..., Studio Monitor, Source Preview, External Previ..., VMIX 1 in 1, VMIX 1 in 2 ON LWPS80236E, VMIX 1 in 3, and VMIX 1 in 4. A yellow vertical bar highlights the 'Studio Monitor ON LWPS80236E' column. A yellow horizontal bar highlights the 'VMIX 1 in 2 ON LWPS80236E' row. The intersection of these two bars is also highlighted. There are navigation arrows and a 'P' button at the bottom right.

	Program 1	Program 2	Program 4 R...	Aux Send 1	Aux Send 2	CR Monitor ...	CR Monitor	CR Headpho...	Preview	Talk to CR	Guest Head...	Studio Monitor ON LWPS80236E	Talent Hdph...	Talkback	Program 3	Program 4	Aux Send 3	Aux Send 4	
Channel 1																			
Channel 10																			
Return 1																			
Return 2																			
CR Monitor																			
CR Headpho...																			
Studio Monitor																			
Source Preview																			
External Previ...																			
VMIX 1 in 1																			
VMIX 1 in 2 ON LWPS80236E																			
VMIX 1 in 3																			
VMIX 1 in 4																			

- **Search bar:** The top of the component contains a search bar. This allows the user to search sources or destinations by typing in either the S or D field to reduce the columns and rows shown. It can be further filtered by using the drop downs to specify a specific device. It is important to note that the device drop downs will not be present on virtual routers. They will only exist on the audio and/or gpio router. The search bar may also be hidden if desired as described in the styling section of this document below.
- **Sources:** Source Names are displayed in the columns at the top of the table. Hovering over any column header will cause a yellow bar to appear highlighting that source and expanding the name of the source to show its full description. Double clicking on a source column header repeatedly will cause the grid to jump to each destination the source is routed to and highlight it with the yellow hover bars. If the source is not routed to any destination, nothing will be highlighted.
- **Destinations:** Destination Names are displayed in the left most column of each row. Hovering over any row header will cause a yellow bar to appear highlighting that destination and expanding the name of the destination to show its full description. Double clicking on a destination row header will

cause the grid to jump to the source that is currently routed to that destination. If no source is routed to the destination, then nothing will be highlighted. Destination names may also be greyed out indicating the destination is locked either at the user level or system level and can therefore not be changed. Double clicking on a locked row header will still navigate to the locked cross-point if a route exists.

- **Routing Grid:** Each cross-point in the grid will have one of several colors indicating the cross-point state:

- The default background color (light gray in the pictures above) indicates no route exists at that cross-point.



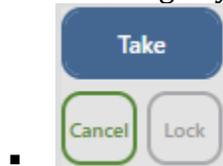
- Blue indicates the route exists at that cross-point. Note that if the cross point is locked there may also be either a system level or user level lock icon on the cross point.



- Red indicates a cross-point that has been preset for an action. Clicking on a cross point will preset it and the action buttons will then become enabled for use.



- To take, clear, or lock a route, click on the desired cross-point to preset it and then click the corresponding Take/Clear or lock button in the top left corner of the grid. See more on the action buttons below. The cancel button can be used to clear all selected presets without making any changes.



- **Action Buttons:** There are 4 action buttons. In the top left corner of the grid is the Take/Clear button, the cancel button, and the lock/unlock button. In the bottom right corner is the preset rotation button. In all cases the button will be grayed out when it is not available for use because no selection (presets) have been made. When a preset is made the border and text of the available button will light up showing it is available for use and hovering over the button will light up its background.

- **Take/Clear:** The Take/Clear button will display either take or clear depending on whether the route that is preset is currently an active route or not. Clicking this button will either clear or take the route.



- **Cancel:** The Cancel button may be used to cancel any presets without making any route changes.



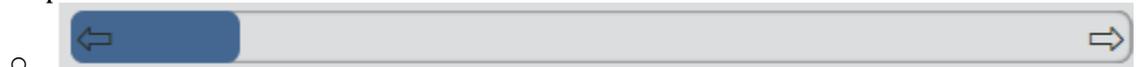
- **Lock:** The lock button will only be available if an existing cross point is selected and may be used to add a user level lock on the route. If the cross point is already user level locked, the lock button will show unlock instead and allow you to unlock the route if you have the correct user rights to do so.



- **Preset Rotate:** The preset rotate button is located in the bottom right corner of the matrix component. When disabled it will show a P. When a preset is selected, it will show the count of selected presets. Click on the button will cause the hover bars to highlight the selection moving the scroll bars to the correct preset cross point if necessary. If the matrix has been configured in multiple preset mode, this button will loop through the currently selected presets.



- **Scroll Bars:** Horizontal and Vertical scroll bars will appear and/or disappear to the right and below the matrix if there are more destinations or sources than can be displayed. The scrollbars may be manipulated by dragging anywhere within the bar or by clicking and/or holding the arrows at either end of the scroll bar. The scroll bars are slightly wider than some normal windows scroll bars because the grid is designed to be highly accessible for touch panel use.



Configuration properties

The Router XY Matrix has many properties that can be changed and manipulated to affect how it works when used in a panel. While there are many properties listed here the three most important as far as functionality changes are concerned are routerpath, matrix-mode, and touch-hover.

- **Style Properties**
 - **routerpath:** Used to select the router which will be used with the matrix.
 - **matrix-mode:** There are three ways you can configure the xy matrix to work.

- **Preset (default):** Clicking a cross-point presets it for a take or clear. Clicking a different preset will clear the previous preset. Clicking a cross-point that has already been preset will remove the preset. Only one preset selection is allowed at a time.
 - **Preset-multiple:** In this mode you can preset multiple routes for a clear or take and then take or clear them all at once. The preset rotator button will allow you to loop through the selected presets.
 - **Single:** In this mode all action buttons are removed, and routes are made just by clicking on a cross point. This is useful for simple non-critical virtual routing matrices. But this mode should not be used for mission critical matrices because it is too easy to accidentally click or touch a cross point and make an inadvertent change.
- **touch-hover:** When this option is set to true and the matrix is being used with a touch screen, the first tap will generate the same hover lines as you see when hovering a mouse over the cross-point. The second touch will preset. If this property is false, the first touch will both set the hover lines and set the preset.
- **destination-search, destination-device-search, source-search, source-device-search:** Each of these correspond to whether that particular search field appears and is available in the search bar. Setting all of them to false will hide the search bar. This is particularly useful for small matrices where searching is not necessary. Please note that even if the device selection drop-down fields are set to true, they will still not appear for virtual routers. They will only appear if used with an audio or gpio router. Virtual routers will display the textual search fields for both source and destination if they are enabled in this property.
- **fixed-col-row-length:** This defined the length of the row and column headers in pixels.
- **row-height:** Defines the height of the rows and columns in the grid. It is important to note that the grid will always be filled. Therefore, if there are not enough entries to fill the grid, these parameters may be larger than this height. Also, if you reduce this parameter you may find the grid extending off the bottom of the component. To correct this, reduce the font-size as well until it fits again. The sizing of items in the grid is a delicate balance so if you start altering these parameters from the defaults, you may have to tweak to get it to display correctly. The height is intentionally large enough for touch use.
- **route-engaged-color:** This is the color (blue by default) of cross-points where a route exists.
- **route-preset-color:** This is this the color (dark red by default) of cross-points that have been preset for an action.

- **route-hover-color:** This is the color of the hover bars that show the row and column the mouse is hovered over.
- **grid-line-color:** This is the color of the lines in the grid.
- **disabled-button-text-color:** This is the color of the text in disabled action buttons.
- **enabled-button-text-color:** This is the color of the text in enabled action buttons. This property is no longer used as the text color for enabled uses the specific button's enabled color below.
- **disabled-button-background-color:** This is the background color of disabled buttons.
- **enabled-button-background-color:** This is the background color of enabled buttons. This is overridden by hover effects.
- **locked-destination-background-color:** This is the color of cross-points that are locked.
- **locked-destination-text-color:** This is the color of the text in the row headers for destinations that are locked.
- **matrix-scroll-bar-height:** This is the height of the horizontal scroll bar and is also used as the width of the vertical scroll bar. Scroll bars will automatically disappear if all cross-points can fit on the matrix.
- **matrix-search-bar-height:** This is the height of the search bar. Please note that the correct way to hide the search bar is to disable the search field properties mentioned earlier in this property list.
- **scroll-bar-color:** This is the color of the active part of the scroll bars.
- **scroll-bar-background-color:** This is the color of the inactive (background) portion of the scroll bars.
- **column-row-header-back-color:** This is the background color of the row and column headers.
- **column-row-text-color:** This is the color of the text for column and row headers.
- **route-hover-text-color:** This is the color of the text for column and row headers when the hover/highlight selection bar is active on that row or column.
- **cancel-enabled-button-color:** This is the color used by the border and text when the cancel button is active as well as the background color of the cancel button if the button is enabled and the mouse is hovering or clicking on it.
- **take-enabled-button-color:** This is the color used by the border and text when the take/clear button is active as well as the background color of the take/clear button if the button is enabled and the mouse is hovering or clicking on it.
- **lock-enabled-button-color:** This is the color used by the border and text when the lock button is active as well as the background color of the lock button if the button is enabled and the mouse is hovering or clicking on it.
- **preset-enabled-button-color:** This is the color used by the border and text when the preset rotation button is active as well as the

background color of the preset rotation button if the button is enabled and the mouse is hovering or clicking on it.

- **disabled-button-border-color:** This is the border color used by disabled action buttons.
- **grid-offset:** This is the number of pixels of offset between the edge of the matrix and the scroll bars.
- **hover-event-timeout:** This is the number of milliseconds that a cross-point must be hovered over before raising the hover events that may be used in logic flows.

- **Action Properties**

- **trigger-take-clear:** This property may be used by a logic flow and/or binding to remotely press the take/clear button. This will only have an effect if there is a cross point preset for action.
- **trigger-lock:** This property may be used by a logic flow and/or binding to remotely press the lock button. This will only have an effect if there is a cross point preset for action.
- **trigger-cancel:** This property may be used by a logic flow and/or binding to remotely press the cancel button. This will only have an effect if there is a cross point preset for action.
- **set-preset:** This property allows a logic flow to preset a cross-point. The syntax involves a string with the source and destination path with the `_X_` between the two. Example:
 - `tcp://172.16.1.97:93?l=SRC&d=src&i=2&t=aaudio_X_tcp://172.16.1.72:93?l=DST&d=dst&i=31&t=aaudio`

- **Events**

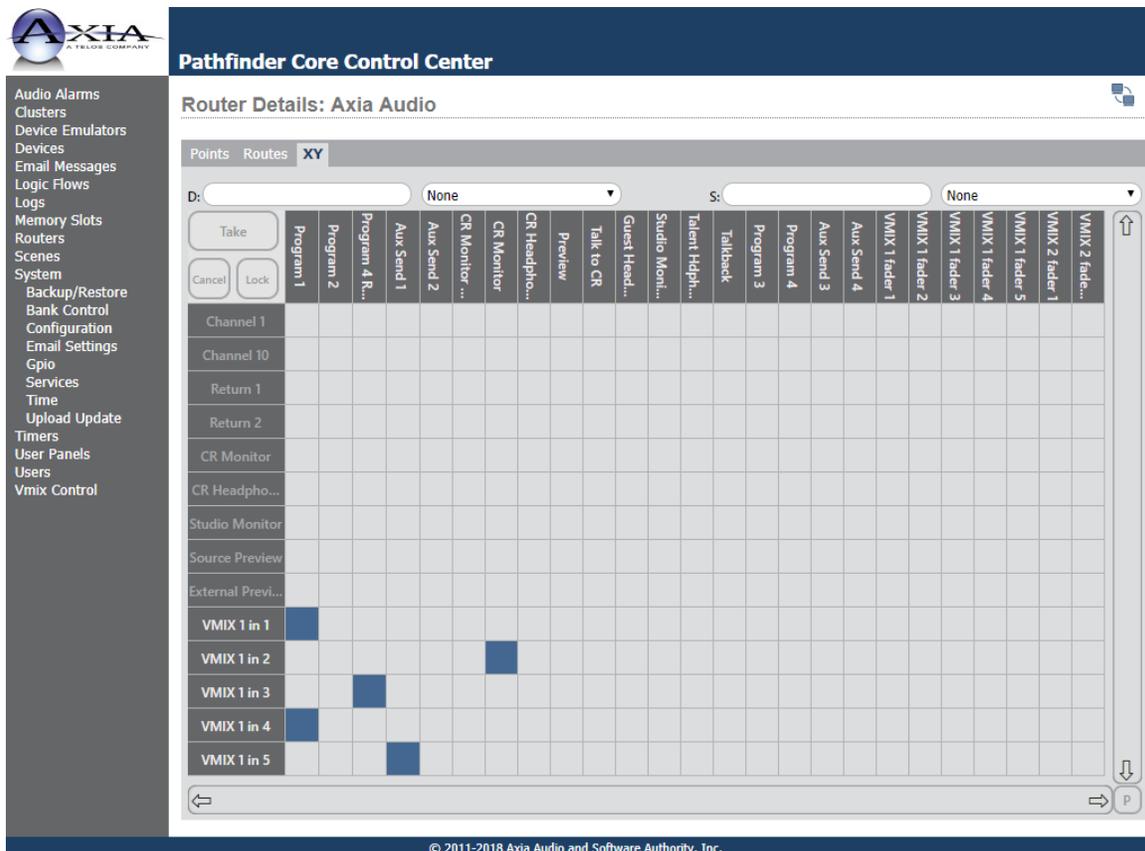
- **hover-source-name:** Event raises the name of the source for the cross-point that the mouse is currently hovering over.
- **hover-source-description:** Event raises the description of the source for the cross-point that the mouse is currently hovering over.
- **hover-source-path:** Event raises the path of the source for the cross-point that the mouse is currently hovering over.
- **hover-destination-name:** Event raises the name of the destination for the cross-point that the mouse is currently hovering over.
- **hover-destination-description:** Event raises the description of the destination for the cross-point that the mouse is currently hovering over.
- **hover-destination-path:** Event raises the path of the destination for the cross-point that the mouse is currently hovering over.
- **preset-source-name:** Event raises the name of the source for the cross-point that has been preset.
- **preset-source-description:** Event raises the description of the source for the cross-point that has been preset.
- **preset-source-path:** Event raises the path of the source for the cross-point that has been preset.

- **preset-destination-name:** Event raises the name of the destination for the cross-point that has been preset.
- **preset-destination-description:** Event raises the description of the destination for the cross-point that has been preset.
- **preset-destination-path:** Event raises the path of the destination for the cross-point that has been preset.

One of the interesting things you can do with the hover and preset events is to bind the path to the io field of meters. This allows you to create a panel where hovering over a cross-point also shows metering for that cross-point. Note that metering does not yet support io binding to virtual router ios.

Router XY Matrix router details web page

In addition to being able to use the Router XY matrix in a user panel, there is also now an XY Matrix tab on the router details web page of each router.



The screenshot displays the 'Router Details: Axia Audio' interface, specifically the 'XY' matrix tab. The interface features a sidebar on the left with various system and configuration options. The main area shows a grid of cross-points between input sources (rows) and output destinations (columns). The input sources listed are Channel 1, Channel 10, Return 1, Return 2, CR Monitor, CR Headpho..., Studio Monitor, Source Preview, External Previ..., VMIX 1 in 1, VMIX 1 in 2, VMIX 1 in 3, VMIX 1 in 4, and VMIX 1 in 5. The output destinations listed are Program 1, Program 2, Program 4 R..., Aux Send 1, Aux Send 2, CR Monitor..., CR Headpho..., Preview, Talk to CR, Guest Head..., Studio Moni..., Talent Heph..., Talkback, Program 3, Program 4, Aux Send 3, Aux Send 4, VMIX 1 Fader 1, VMIX 1 Fader 2, VMIX 1 Fader 3, VMIX 1 Fader 4, VMIX 1 Fader 5, and VMIX 2 fader... The grid shows blue squares indicating active connections: VMIX 1 in 1 to Program 1, VMIX 1 in 2 to CR Headpho..., VMIX 1 in 3 to Program 4 R..., VMIX 1 in 4 to Program 1, and VMIX 1 in 5 to Aux Send 2. The interface includes a sidebar with navigation options, a top header with the Axia logo, and a footer with the copyright notice '© 2011-2018 Axia Audio and Software Authority, Inc.'

It functions exactly like the user panel matrices except with all properties in their default states.

Version 1.5.20.29 Changes

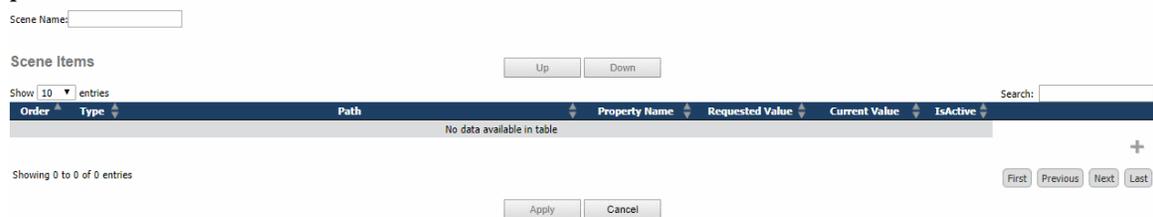
Scene Editing

This version introduces a web page for creating and editing scenes. It is important to understand that scenes in PathfinderCore PRO are nothing more than a list of property changes. They could be a list of route changes but they could also be a list of memory slot changes, a list of vmix changes, or any combination of the above. Any property that is available in the system can be a part of a scene. To view the available scenes in the system, click the scene navigation menu item.

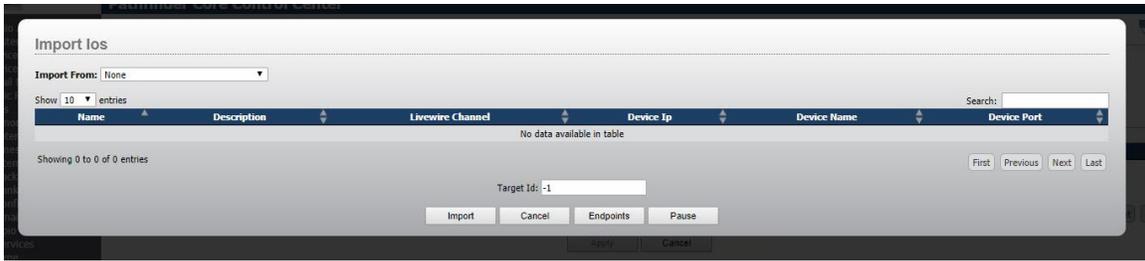


The activate button will cause all items in the scene to be set to their defined state. The “IsActive” field displays whether all items in the scene are in their requested state. This property can also be used in logic flows to take other actions. The clone link will create a copy of the scene. This is useful if you need to create multiple scenes with the same items but different values for each item. The edit link will open the editor and display the data for an existing scene. The minus icon will delete the scene. Clicking the plus icon will open the editor with a new blank scene.

To create a new scene, select the scenes navigation menu item and then click the plus icon.

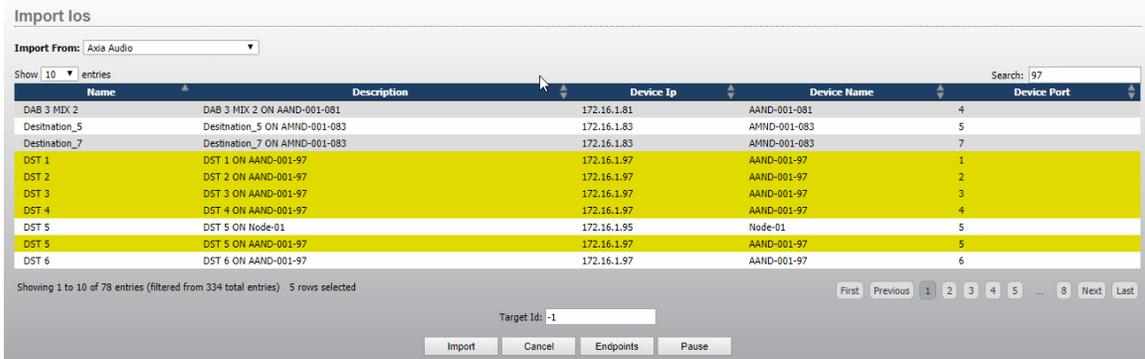


Each scene must have a unique name and a scene may not be saved without a name. Use the name field to create a name for the scene. Use the plus icon to add items to the scene.

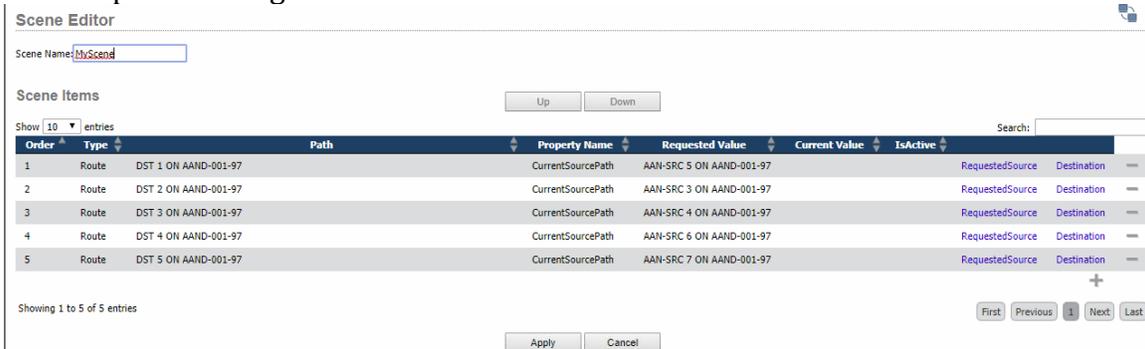


Route point scene items

By default the scene addition dialog will open with a route destination import dialog. Select the router from the import from list and then select the destinations to import. Use the shift key to select multiple destinations.



Items from different routers may also be a part of the same scene by clicking the plus icon again and selecting ios from a different router in the routers drop down. Click import to bring the ios into the scene.

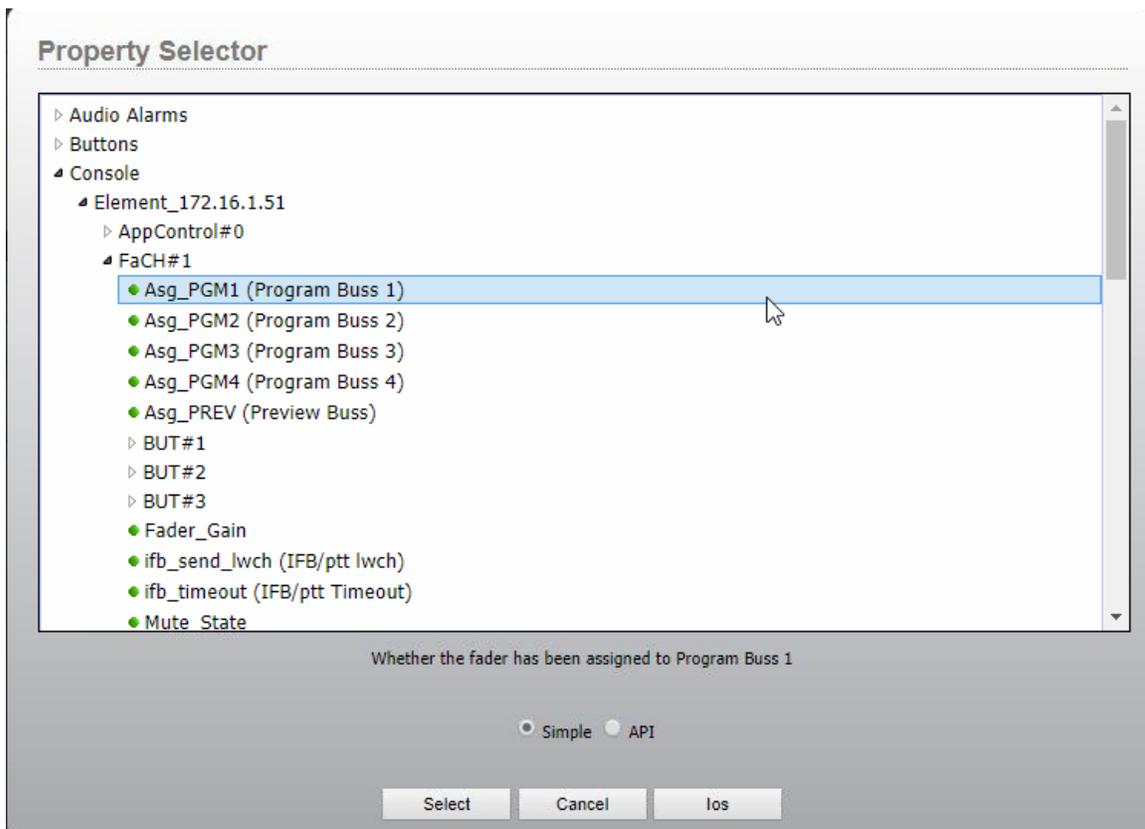


The scene editor will import the route destinations into the scene. The requested value reflects the source that will be taken if that scene item is executed. By default it will be set to the current source that is routed to the destination. This can be changed by clicking the RequestedSource link. This will open a list of sources that are present on the selected router so that you can select a different source as the value for the route point in the scene item. Clicking the destination will allow you to select a different destination for the scene item if you selected the wrong one and need to change it. Clicking the minus icon will delete the item from the scene. At

this point the scene still does not exist. You have to click apply to save the changes and return to the scene list. At that point the scene will be available for use. This is important to remember. Changes made while editing a scene will not actually be stored and become usable until the changes have been applied.

Property scene items

In addition to route points any property in the system can be a scene item. After clicking the add button to add a new scene item to a scene you will also find an endpoints button to switch the route destination dialog to a property selection dialog.



This is the same dialog used to select an endpoint in logic flows. In the example above we are adding a program buss assignment property on an Element console to the scene. The Ios button will return to the route selection dialog. The select button will select the property and import it as a scene item into the scene. The system will remember whether the last item imported during a given scene editing session was a property or route point and return to the last used dialog. You can switch back

and forth using the Endpoints and IOs buttons from the corresponding dialog. After importing a property it will have slightly different links in the scene itself.

Order	Type	Path	Property Name	Requested Value	Current Value	IsActive	RequestedSource	Destination
1	Route	DST 1 ON AAND-001-97	CurrentSourcePath	AAN-SRC 5 ON AAND-001-97			RequestedSource	Destination
2	Route	DST 2 ON AAND-001-97	CurrentSourcePath	AAN-SRC 3 ON AAND-001-97			RequestedSource	Destination
3	Route	DST 3 ON AAND-001-97	CurrentSourcePath	AAN-SRC 4 ON AAND-001-97			RequestedSource	Destination
4	Route	DST 4 ON AAND-001-97	CurrentSourcePath	AAN-SRC 6 ON AAND-001-97			RequestedSource	Destination
5	Route	DST 5 ON AAND-001-97	CurrentSourcePath	AAN-SRC 7 ON AAND-001-97			RequestedSource	Destination
6	Property	Element => FaCH#1	Asg_PGM1	ON			RequestedValue	Property

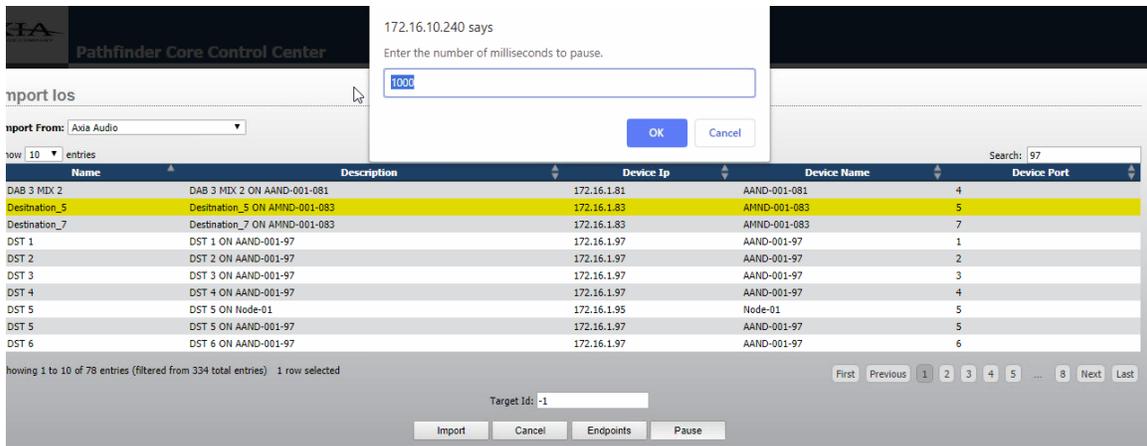
In this case you will see a RequestedValue link and a Property link. The property link will allow you to change the property you selected if you selected the wrong one. By default, the value that will be used for the property item is set to whatever it is at the time the scene item is created. This can be changed by clicking the RequestedValue link.



This will open a value selector. The value selector may be different depending on the type of property you are manipulating. For example if the item is a color property, it might present a color selection dialog. If the property requires text, a text box may be presented. Or it could present a drop down of possible values. In the case above, the program buss assignment could either be on or off, so select the option you want the scene item to set. Note this does not change the current state of the console in the system unless the scene is activated. At this point you are only editing what would happen if you activated the scene.

Order and pause

The scene editor also has a “move up” and “move down” button. These are used to organize the order in which a scene sends its messages to the equipment when activated. It is very important to understand that the order is generally irrelevant because a scene execution does not wait for a change to complete before sending the next scene item change message. They are usually sent as a block of changes and so the actual changes may not happen in the equipment in the same order in which they were sent to the equipment. If the execution order is important it is possible to add pause items into the scene. This will cause the scene to pause as it iterates through the items for a number of milliseconds. To add a pause, click the plus icon and then from the route Ios dialog, click the pause button.



The system will ask for the number of milliseconds to pause and after clicking OK, a pause item will be added to the scene.

Scene Items

Up Down

Show 10 entries

Order	Type	Path	Property Name	Requested Value	Current Value	IsActive
1	Route	DST 1 ON AAND-001-97	CurrentSourcePath	AAN-SRC 5 ON AAND-001-97	RequestedSource	Destination
2	Route	DST 2 ON AAND-001-97	CurrentSourcePath	AAN-SRC 3 ON AAND-001-97	RequestedSource	Destination
3	Route	DST 3 ON AAND-001-97	CurrentSourcePath	AAN-SRC 4 ON AAND-001-97	RequestedSource	Destination
4	Route	DST 4 ON AAND-001-97	CurrentSourcePath	AAN-SRC 6 ON AAND-001-97	RequestedSource	Destination
5	Route	DST 5 ON AAND-001-97	CurrentSourcePath	AAN-SRC 7 ON AAND-001-97	RequestedSource	Destination
6	Property	Element => FaCH#1	Asg_PGHI	ON	RequestedValue	Property
7	Pause	.	Pause	1000		

Selecting the pause item (or any scene item in the scene) and then using the Up and Down arrows will allow you to manipulate the order used to send the scene messages and to manipulate where pauses occur in the scene execution.

Current Scene State and Troubleshooting

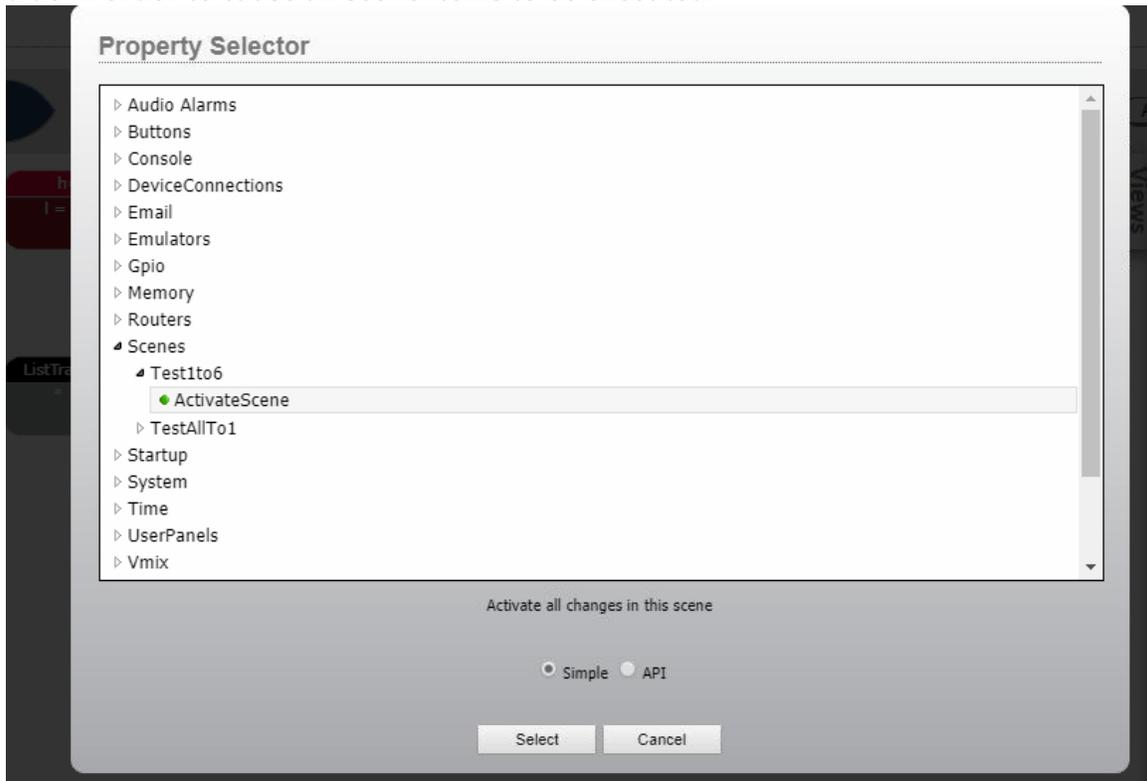
Once a scene has been created the list of scenes will show whether the scene is active or not in the "IsActive" field. If you expect a scene to be active and it is not, you can see what items failed to change by editing the scene.

Property Name	Requested Value	Current Value	IsActive
CurrentSourcePath	AAN-SRC 1 ON AAND-001-97	AAN-SRC 5 ON AAND-001-97	False
CurrentSourcePath	AAN-SRC 2 ON AAND-001-97	AAN-SRC 3 ON AAND-001-97	False
CurrentSourcePath	AAN-SRC 3 ON AAND-001-97	AAN-SRC 4 ON AAND-001-97	False
CurrentSourcePath	AAN-SRC 4 ON AAND-001-97	AAN-SRC 4 ON AAND-001-97	True
CurrentSourcePath	AAN-SRC 5 ON AAND-001-97	AAN-SRC 5 ON AAND-001-97	True
CurrentSourcePath	AAN-SRC 6 ON AAND-001-97	AAN-SRC 3 ON AAND-001-97	False

The requested value will show the expected value for each scene item in order for it to be considered active and the current value will show what the actual value of the scene item is. Note: there is currently an issue that the currentvalue data is not updated without exiting and re-entering the scene edit. In the example above we can see that two of the scene items are in the correct state to make the scene active and the other two are not.

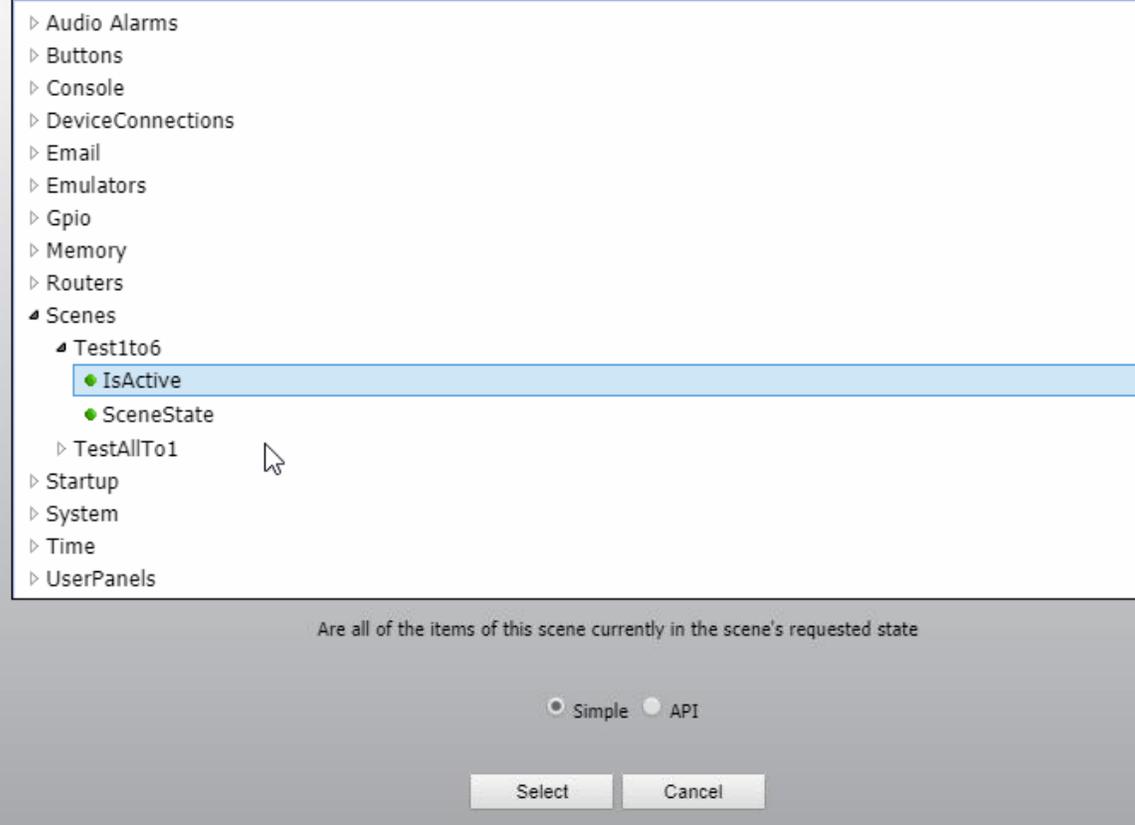
Scenes and Logic Flows

There are a number of properties that may be used with scenes in logic flows. When an endpoint is selected each scene has an ActivateScene property that can be set to true in order to cause all scene items to be executed.



As a Start point, two properties are available: IsActive and SceneState.

Property Selector



The IsActive property will either be true or false depending on whether all items in the scene are currently at their requested value. The SceneState property extends this slightly with three options: All, Some, Or None. This will change depending on whether all items have their requested values, some do, or none do.

Version 1.5.20.31 Changes

Generic Device Emulators

Watcher TriggeredValue property

Generic Device Emulator Watchers now have an additional momentary property called TriggeredValue. When the watcher input is discovered that value will be assigned to the TriggeredValue property momentarily and then the value will be set back to blank. It is momentary in order to raise changes when the same value enters the emulator repeatedly. This is most useful in combination with the regular expression option below.

Watchers and Regular Expressions

Generic Device Emulator Watchers now support the use of regular expressions (also called regexes). Regular expressions are an advanced language used for pattern matching in textual information. Specifically, we use the Microsoft dotnet variation of regular expressions. Regular expressions are beyond the scope of this document, but more can be learned using the links below:

<https://www.regular-expressions.info/>

<https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference>

<http://regexstorm.net/tester>

To use a regular expression in a Watcher, use either `Regex.Match(<expression>)` or `Regex.IsMatch(<expression>)` for the value of the watcher. For example:

CBS050

We could use a regex such as this:

```
Regex.Match((?<=CBS).{3})
```

This regex would cause the watcher to fire any time it sees the 6 characters where the first three are CBS and it will put the three characters that fall after the CBS into the TriggeredValue property. This expression makes use of the lookback feature of regular expressions.

A different watcher could then be created for a different show from the same emulator.

```
Regex.Match((?<=NBC).{3})
```

In addition, you could use `Regex.IsMatch()`. This does the same thing except the TriggeredValue will output true if a match exists rather than the value of the match. So, this option is largely redundant to the triggered property and will be more useful in logic flows as discussed below.

It is important to factor in carriage returns and linefeeds when thinking about pattern matching if the incoming data is a line-based protocol. Incorrectly considering the existence or lack of existence of these characters when parsing incoming data will cause incorrect results. For example `(.*)` matches all characters except a line feed. If the protocol uses carriage returns and line feeds at the end of each message, you could use something like `(.*\r\n)` or `((?s).*)`. The latter uses single line mode as described below.

Warning: Regular Expressions are extremely advanced and can appear to be very complicated. One joke about regular expressions is that once you solve a problem with a regular expression you now have two problems. However, they can also be awesomely powerful for the situations where they are needed. It is one of the most commonly used and advanced text pattern matching and manipulation languages across all programming languages used today. However, they will also be slower and more cpu intensive than the normal matches Pathfinder does because they have not been optimized at compile time in the same way as the native code. Therefore, they should be used with caution and (like seasoning) sparingly. Use it when you need it and not where Pathfinder Core PRO's inherent pattern matching is a better option. Also test thoroughly. It is quite easy to design a regular expression that you think is correct but misses certain edge cases. The regex tester listed in the regex links above can be very useful in this case.

Also note that there are a number of options that can be used within the expression to cause the pattern matcher to function differently. For example, by default the dot

character "." matches any character except the new line. If you want it to match the new line character as well, you can turn on the single line option.
Regex(?:s).*).

In this case the inline (?s) tells the regular expression engine to use the single line option when analyzing. The options can be found at:
<https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-options>

Again, this is an advanced feature and incorrect use can cause unpredictable results. But it is also a very powerful tool for certain situations.

Logic Flows

Translation Advanced Field for Conversion List Item Editing

This version adds a field under the advanced link of the translation dialog that can be used to manually enter the data for a conversion item. It is called the Advanced Conversion field. This allow advanced users to bypass some of the helper dialogs (like the color selections dialog) and enter the conversion item entry directly. It is important to note that incorrect data entry into this field can cause unpredictable results.

To use the field, select a conversion list item and then click the Advanced link.

The image shows a screenshot of the "Translator Properties" dialog box. At the top, the title "Translator Properties" is displayed. Below the title is a large text area containing the text "I=True". Below the text area is a row of buttons: "<", "Add", "Insert Before", "Insert After", "Remove", and ">". Below these buttons is a row of two dropdown menus. The first dropdown menu is set to "Low" and the second is set to "True", with an "=" symbol between them. Below the dropdown menus is a link labeled "Advanced". Below the "Advanced" link is a text input field labeled "Advanced Conversion:" containing the text "I=True". Below the text input field is a checkbox labeled "Skip startup state request and wait for next change." which is currently unchecked. Below the checkbox is a text input field labeled "Recursion detection settings:" containing the text "50/1000". At the bottom of the dialog box, there are three buttons: "PinState=h", "Done", and "Cancel". In the bottom right corner, the text "SetTimeStamp" is visible.

Enter the correct Conversion text into the advanced field. The syntax is <FromValue>=<ToValue>. Example:

```
l=True
```

If the From or True value needs to include an equals sign, then enclose the value in Quotes.

```
"abc=frs"]=True
```

In this case the from value will be abc=frz. It is also very important to note that the helper dialogs attempt to present friendly information which may or may not be the underlying value needed in the conversion. For example, when using the current source path property, the conversion list and normal drop-down lists will show you the friendly name of the source. But the actual conversion list item needs the uri based pathio which looks something like:

```
"tcp://172.16.1.53:93?l=SRC&d=src&i=6&t=aaudio"
```

If you are uncertain, create an example of what you are trying to do using the normal conversion list selection methods and then select it to view in the advanced field the actual values that need to be applied. If you have questions, contact support. Again, this is an advanced option for certain unique scenarios.

Translation <NoChange> option

If the value <NoChange> is applied to the “to” portion of a translation, then the output of the translator will not be changed. For example:

```
AAA=True  
BBB=<NoChange>  
*=False
```

In this case an input value of AAA will result in True, BBB not cause anything to change on the output of the translator, and anything else will result in False. In many cases a NoChange is not required because you can just not supply the input and then nothing will happen if that input occurs. But in situation where you are using the * as a catchall but want to explicitly exclude a value from making a change, this can be used. The drop-down lists for possible To values should now include this and it can also be manually entered for a conversion item in situations where the property is textual. In cases where it is a color or numeric property type, you can use the Advanced field described above to enter the <NoChange> value for the conversion list item. <NoChange> is not applicable to the “from” side of a translation and will be interpreted literally if entered on the “from” side.

Translation * in to conversion list items

This is not a new feature but we have realized that it was not well documented. It is common to have a conversion list item in a translator of

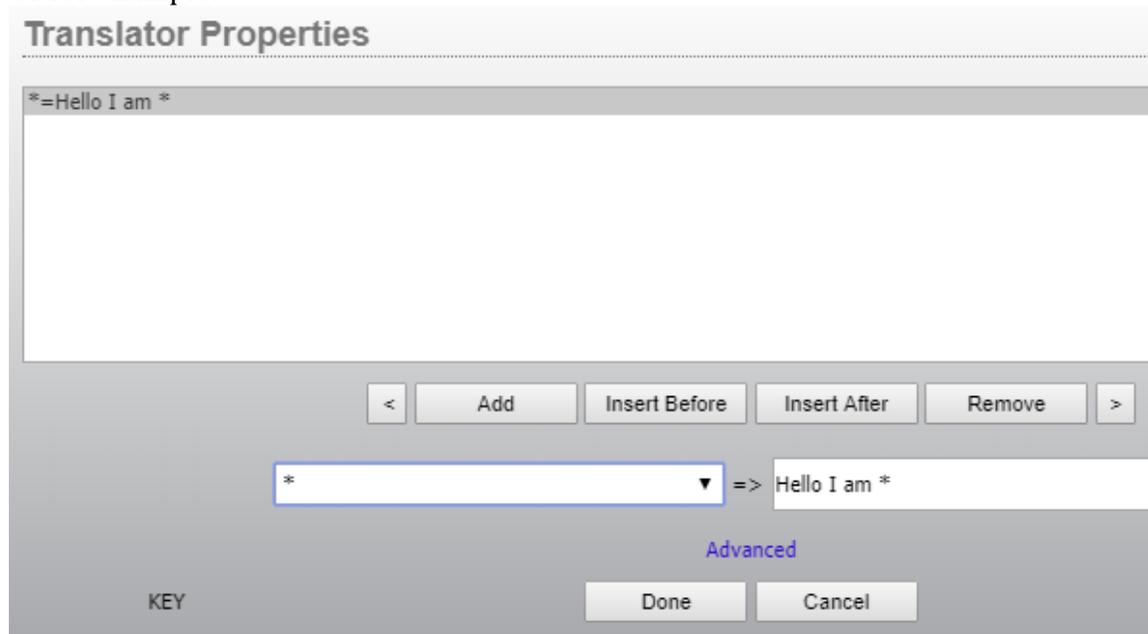
`*=*`

But it is not well known that the * on the “to” side does not need to be by itself. For example, you could have:

`*=Hello * My name is`

In this case the input value would replace the * in the output value. So, it might end up outputting: Hello Dan My name is

Or for example:



This can provide similar functionality natively in the flow translation as string builder memory slots can. * cannot be used in the middle of data on the from side. Use regular expressions to do complex matching if needed as described below.

Translation Regular Expressions

Regular expressions as described in the device emulator watchers above can also now be used in the From portion of a conversion list item. In general, you will use the Advanced conversion field described above to achieve this functionality:

Translator Properties

Regex.IsMatch(-.*)=OFF
*=ON

< Add Insert Before Insert After Remove >

* [input field] => OFF

Advanced

Advanced Conversion: Regex.IsMatch(-.*)=OFF

Skip startup state request and wait for next change.

Recursion detection settings: 900/1000

Fader_Gain="-3276.8" Done Cancel Value

In this case we have created a flow where fader gain values that are not negative will result in a button turning on and otherwise the button indicator will be off. Because fader gain generates lots of data as it slides we have also increased the recursion detection to prevent this flow from getting disabled.

We are using the IsMatch property here so the conversion list item will be used if the value applied is a match in the regex. The use of regular expressions in translations is very similar to what we have described above in the generic emulator section so review that section for details. Again it is extremely important to be aware of the warnings listed above in the discussions of regexes as they apply here as well. They can be slower and more cpu intensive. Use them only where needed and use the normal translation elsewhere.

Sap Property Router

This version introduces a new router type called SapProperty router. The basic concept is that any property that can be read from in PathfinderCore PRO can become a router source and any property that can be written to can become a router destination. Perhaps an easy example might be the GpioByPin router from the Legacy PathfinderPro. It was possible to create a router of individual gpio pins instead of routing entire ports. This can be accomplished in a SapPropertyRouter in PathfinderCore PRO by browsing to the Routers navigation link and clicking the plus icon to create a new router and then selecting the SapPropertyRouter from the available router options.

Add Router

Router Type

Router Id A value of 0 will auto generate an id.

Router Name

Router Description

Then click Add and the new router will be created. This router does not yet have any ios in it.

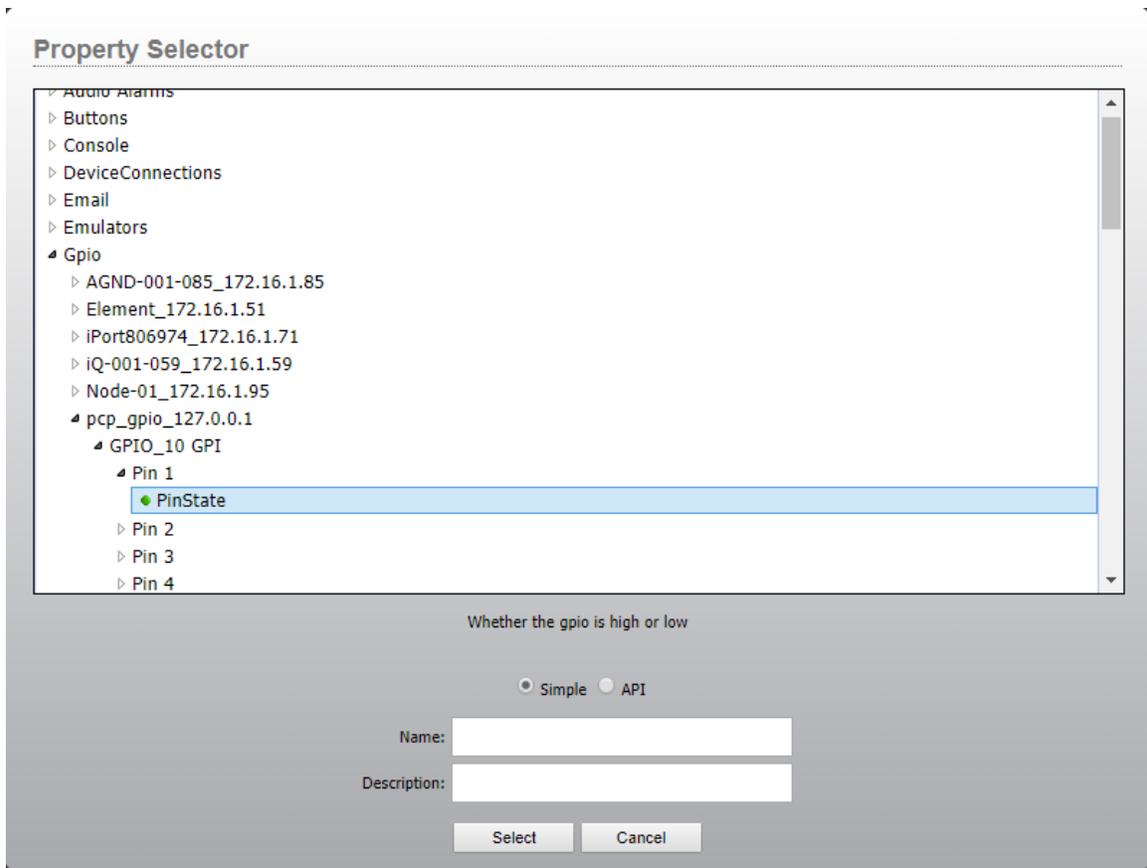
Routers

Show 10 entries

Id	Name	Description	Type	Created	
1	Axia Audio	Axia Audio Router	AxiaAudio		details
2	Axia GPIO	Axia Gpio Router	AxiaGPIO		details
3	MyVirtual		<input type="radio"/> Virtual		details --
4	NewVirt		<input type="radio"/> Virtual		details --
5	MySapRouter		SapProperty		details --
6	DanPinExample		SapProperty		details --

Showing 1 to 6 of 6 entries

Click the details link and then click the points tab. Similar to the virtual router, you will see an import sources and import destinations button as well as a new Translation button. However, clicking the import sources or import destinations button for this router will display the property selector used in logic flows.



In this case we are going to select a gpi pin state property. Provide a name and description for the IO (or allow it to use a default one and click Select). Repeat for other pins you want as sources. Then import the destinations.

Property Selector

- ▷ AGND-001-085_172.16.1.85
- ▷ Element_172.16.1.51
- ▷ iPort806974_172.16.1.71
- ▷ iQ-001-059_172.16.1.59
- ▷ Node-01_172.16.1.95
- ▾ pcg_gpio_127.0.0.1
 - ▷ GPIO_10 GPI
 - ▾ GPIO_10 GPO
 - ▾ Pin 1
 - PinState
 - PulseHigh
 - PulseLow
 - ▷ Pin 2
 - ▷ Pin 3
 - ▷ Pin 4
 - ▷ Pin 5
 - ▷ GPIO_11 GPI
 - ▷ GPIO_11 GPO

Whether the gpio is high or low

Simple API

Name:

Description:

Note that if the ports you are using for destinations are on a livewire driver or Pathfinder Core PRO gpio node then you could use GPI pins as the destination and route gpio to gpio. When done it might look something like this.

Router Details: DanPinExample

Points Routes XY

Import Sources Import Destinations Translation

Show 10 entries

Availability	Name	Source	Description	IP Address	Host	Name	Port
Available	GpioPinState#1 PinState	GpioPinState#1 PinState	GpioPinState#1 PinState	127.0.0.1:9600	PathfinderCore PRO		1
Available	GpioPinState#2 PinState	GpioPinState#2 PinState	GpioPinState#2 PinState	127.0.0.1:9600	PathfinderCore PRO		2
Available	GpioPinState#3 PinState	GpioPinState#3 PinState	GpioPinState#3 PinState	127.0.0.1:9600	PathfinderCore PRO		3
Available	GpioPinState#4 PinState	GpioPinState#4 PinState	GpioPinState#4 PinState	127.0.0.1:9600	PathfinderCore PRO		4
Available	GpioPinState#5 PinState	GpioPinState#5 PinState	GpioPinState#5 PinState	127.0.0.1:9600	PathfinderCore PRO		5

Showing 1 to 5 of 5 entries

Show 10 entries

Availability	Name	Destination	Description	IP Address	Host	Name	Port
Available	GpioPinState#1 PinState	GpioPinState#1 PinState	GpioPinState#1 PinState	127.0.0.1:9600	PathfinderCore PRO		1
Available	GpioPinState#2 PinState	GpioPinState#2 PinState	GpioPinState#2 PinState	127.0.0.1:9600	PathfinderCore PRO		2
Available	GpioPinState#3 PinState	GpioPinState#3 PinState	GpioPinState#3 PinState	127.0.0.1:9600	PathfinderCore PRO		3
Available	GpioPinState#4 PinState	GpioPinState#4 PinState	GpioPinState#4 PinState	127.0.0.1:9600	PathfinderCore PRO		4
Available	GpioPinState#5 PinState	GpioPinState#5 PinState	GpioPinState#5 PinState	127.0.0.1:9600	PathfinderCore PRO		5

Showing 1 to 5 of 5 entries

Now if you go to the routes tab you will see the destinations with a take table of the sources and you can make route changes.

Source				Destination				
Name	Description	IO	Host Name	Name	Description	IO	Host Name	Lock
GpioPinState#5 PinState	GpioPinState#5 PinState		PathfinderCore PRO	GpioPinState#1 PinState	GpioPinState#1 PinState	1	PathfinderCore PRO	
				GpioPinState#2 PinState	GpioPinState#2 PinState	2	PathfinderCore PRO	
				GpioPinState#3 PinState	GpioPinState#3 PinState	3	PathfinderCore PRO	
				GpioPinState#4 PinState	GpioPinState#4 PinState	4	PathfinderCore PRO	
				GpioPinState#5 PinState	GpioPinState#5 PinState	5	PathfinderCore PRO	

Here I have routed GPI pin 5 to GPO pin 1. Therefore, any closures that take place on pin 5 will be replicated on pin 1.

If we click back to the points tab you will also find a Translation button. Clicking on that button will open the translation pattern used for any values that pass from the source property to routed destination properties.

Translator Properties

< Add Insert Before Insert After Remove >

* => *

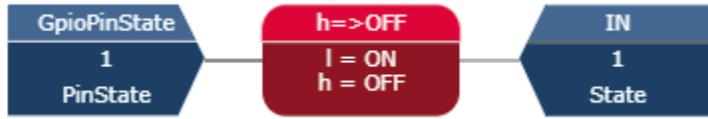
Advanced

PinState Done Cancel PinState

There is a single translation pattern that is used between the source values and destination values. In this case it is *=* and therefore a pin low on the source side will cause a pin low on all destinations that source is routed to. We could change this translation to be l=h and h=l such that lows on the source would generate high to any destination that source was routed to. Another interesting option might be to change the output properties to be pulse properties rather than pin state properties. Then you could make the translation be l=5000. This would cause the output to go low for 5000 ms each time the input went low and nothing to happen when the input went high.

If you have been using logic flows for some time you might be wondering why we need this. We could do all of this with flows. The usefulness of doing it in a router comes into play when what would be the start point of a flow needs to change dynamically. Another example might make this clearer. Suppose we have a vmixer assigned to each air chain. Channel 1 of the vmixer represents the feed from the

studio console. There is a gpi pin in the studio which turns the vmix channel on. This is simple to accomplish with a logic flow:



However, what if we have 5 studios and any studio could be assigned to the air chain. Now in logic flows this becomes much more complex because we have to build flows for each variation of studio to air chain possibility. However, this becomes easy with a property router. Again, on the source side of the router we select GPI pin state properties. But on the destination side we pick all of the air chain vmix IN state properties.

Points Routes XY

Import Sources Import Destinations Translation

how 10 entries Search:

Availability	Name	Source	Description	IP Address	Host	Name	Port
Available	GpioPinState#1 PinState	GpioPinState#1 PinState	GpioPinState#1 PinState	127.0.0.1:9600	PathfinderCore PRO		1
Available	GpioPinState#2 PinState	GpioPinState#2 PinState	GpioPinState#2 PinState	127.0.0.1:9600	PathfinderCore PRO		2
Available	GpioPinState#3 PinState	GpioPinState#3 PinState	GpioPinState#3 PinState	127.0.0.1:9600	PathfinderCore PRO		3
Available	GpioPinState#4 PinState	GpioPinState#4 PinState	GpioPinState#4 PinState	127.0.0.1:9600	PathfinderCore PRO		4
Available	GpioPinState#5 PinState	GpioPinState#5 PinState	GpioPinState#5 PinState	127.0.0.1:9600	PathfinderCore PRO		5

Showing 1 to 5 of 5 entries

how 10 entries Search:

Availability	Name	Destination	Description	IP Address	Host	Name	Port
Available	IN#1 State	IN#1 State	IN#1 State	127.0.0.1:9600	PathfinderCore PRO		1
Available	IN#2 State	IN#2 State	IN#2 State	127.0.0.1:9600	PathfinderCore PRO		2
Available	IN#3 State	IN#3 State	IN#3 State	127.0.0.1:9600	PathfinderCore PRO		3
Available	IN#4 State	IN#4 State	IN#4 State	127.0.0.1:9600	PathfinderCore PRO		4
Available	IN#5 State	IN#5 State	IN#5 State	127.0.0.1:9600	PathfinderCore PRO		5

Showing 1 to 5 of 5 entries

Once the items are selected, we can change the translation to be:

Translator Properties

l=ON
h=OFF

Now when we route a studio pin to the airchain vmix, the low from the studio will turn on and off the vmix state of whichever airchains that studio happens to be routed to.

We can then extend this example by using a virtual router where the virtual source package includes the audio console program buss audio source and the pin source from the SapProperty router and the destination package includes the vmix channel input audio destination and the vmix state property from the property router. The signals get married together such that a single virtual route change will switch both audio and signaling seamlessly. And that virtual router can then be controlled by a

third-party application using something like Probel without that application needing to know that we are actually switching numerous things under the hood. This opens up a whole new world of what can be routed.

Another interesting example uses many of the new features described above. What if we wanted to route a tcp stream of song data to different destinations depending on what is currently on the air. We could create generic device emulators for each destination and each song data source. Using the regex capability above we could add a watcher to the emulators on the source side that looks like:

```
Regex.Match((?s).*)
```

This will match any data coming in. Then we create a new property router and for the sources we choose the watcher TranslationValues. For the destinations we select the ToSend property of the destination generic emulators. This will essentially pass any data that comes in the source emulator to the output of destinations that source is routed to. Then we could marry those points as well into the air chain virtual router such that audio, studio pin to vmix state, and songdata are all married together as a single virtual route. See the 1.4 documentation if you are unfamiliar with combining base sources and destinations into a single package in virtual routers.

Another possibility might be to route memory slots. Or we could use the duration regex example above to capture different satellite duration data as source points in a router that then gets passed on to countdown clocks. The point is you can now route anything Pathfinder knows about.

A few words of caution. In all of the examples above we made all of the sources in a given property router the same type; all gpios, or all watcher TranslationValue properties. And all of the destinations in a specific property router were of a specific type as well; all gpios or all vmix states or all generic emulator tosend properties. While this is not strictly required, it is usually good practice. Once you start mixing io property types in the same router's sources or destinations, the translation table can become non-intuitive and it is possible to output the wrong data to devices. PathfinderCore PRO does have some validation steps to prevent some of that but the possible property list is so vast that we might not catch all invalid values. Keep your signaling layers separate in separate property routers and then marry them together as layers in a virtual router.

Additionally, obviously there is no route state in the equipment for this type of routing. PathfinderCore PRO is doing the lifting to accomplish it. This means that in order to retain the route state between restarts, this router has to store all route changes (not the values transitioning through the routes) to the backing storage. In the case of the fanless engine which uses compact flash for this storage, we have implemented some protections to reduce the write cycles. But it is not recommended to make recursive millisecond route changes on Property routers if it can at all be avoided; for example a rotation circuit that continuously loops through

changing property routes. This is less of a concern with the R2 platform which uses SSD and the vm version.

All of the new features in this build are very much beta features. So please report any issues you might have so that we can investigate.