Developer: Christopher McNeely
Source: https://github.com/professoraire/ElegantXML-Crestron

Elegant XML – Serial Property Interlock v1.4

GENERAL INFORMATION

SIMPL WINDOWS NAME	Elegant XML – Serial Property Interlock v1.4
CATEGORY	Elegant Modules
VERSION	1.4
SUMMARY	Provides a connection point for setting and retrieving Serial values from an XML file.
GENERAL NOTES	This module works with the Elegant XML – Manager module and there must be a Manager module in the program for this module to work.
CRESTRON HARDWARE REQUIRED	3-Series Processor

PARAMETERS

ManagerID	S	The ID of the Manager instance to associate this module with. The Manager instance will control what file is being loaded from.
XmlPath	S	The path to the XML attribute to read from the XML file. The Property Interlock module only reads/writes to a SINGLE property element in the file.
PropertyValue[#] (1-50)	S	The string that the XML attribute will be set to when this input is selected.

CONTROL

Select[#] (1 – 50)	D	Pulse a Select input to set the XML attribute specified by the XmlPath to the associated PropertyValue. The associated IsSelected output will go high indicating the selected value.
SetValue	S	Sets the serial value of the property. If the value matches any of the provided values the relevant SelectedValue output will go high, otherwise all outputs will be low.

FEEDBACK

IsInitialized	D	Transitions high when the module has successfully registered with its Manager module.
IsSelected[#] (1 – 50)	D	Each output goes high when the value of the XML attribute

		specified by the XmlPath is equal to the associated PropertyValue. Note that if you have multiple PropertyValue parameters that are equal only the first will go high, making this a true one-output-at-a-time interlock.
SelectedValue	S	Provides the currently selected value as a string.

ADDITIONAL DATA

Revision History	v1.4 – Initial Release

Additional Details	**Use Notes**
	This module is generally intended for when you would pull a value out of a configuration file and then immediately pass it through an SIO to determine what the value is. The module functions as a digital interlock, reducing the number of modules required to attain similar functionality.
	
	Although the parameter is only capable of being a string value, this doesn't preclude you from placing numeric values on the path, though they won't be interpreted as numeric by the module and processor engine.
	
	Path Notes
	• You do NOT need to place the RootElement on the path. This helps shorten the length of the paths needed.
	• Also note the last optional delimiter in the path. If you place this last delimiter on the path you will be telling the parser to read or write the value inside of a pair of tags, the opening and closing tags. (Like: <Quantity>2</Quantity>) This is useful for XML files that use unique individual elements for all properties, instead of placing multiple values on elements.
	• If a value doesn't exist in the XML file, the value of the corresponding input is used. So, if you time your system startup to initialize the inputs of all your element module and then load the file, if any value isn't found it will use that initialized value instead. This can be useful when creating standardized systems where you'll likely want to start with the same values, or where you want to be able to create XML files with the default values on the fly.
	
	The exact format for specifying paths is as follows: Element[Identifier]{PathDelimiter}Element[Identifier]{PathDelimiter}Element[Identifier][{PathDelimiter}][{DefaultValueDelimiter}DefaultValue]
	
	The following file snippet and paths demonstrate how this works:
	
	`<?xml version="1.0" encoding="UTF-8" ?>`
	`<Config>`
	`  <Switchers>`
	`  <Quantity>2</Quantity>`
	`    <Switcher Id="1">`
	`      <Inputs>`
	`        <Input Id="1" Name="Input 1" Patch="1" IsInstalled="true" />`
	`        <Input Id="2" Name="Input 2" Patch="2" IsInstalled="false" />`
	`      </Inputs>`
	`    </Switcher>`
	`    <Switcher Id="2">`
	`      <Inputs>`
	`        <Input Id="1" Name="Laptop" Patch="9" IsInstalled="false" />`
	`      </Inputs>`
	`    </Switcher>`

```
  </Switchers>
</Config>
```

Paths

1. Switchers.Switcher Id=”1”.Inputs.Input Id=”1”.Name|DefaultName
    a. This outputs “Input 1” if the file is present, or “DefaultName” if not.
2. Switchers.Switcher Id=”2”.Inputs.Input Id=”1”.Patch|1
    a. This outputs “9” if the file is present, or “1” if not.
3. Switchers.Switcher Id=”1”.Inputs.Input Id=”1”.IsInstalled|true
    a. This would return “true” if the file is present, or “true” if not.
4. Switchers.Quantity.|0
    a. This outputs “2” if the file is present, or “0” if not.

You can see in this example that the optional identifier MUST occur immediately following the name of the element. For example, if you attempted to write the path: Switchers.Switcher Id=”1”.Inputs.Input Name=”Laptop” this would fail, since the Name=”Laptop” attribute isn’t immediately following the name of the element, which is Input.

You can also see that the identifier isn’t required. Be wary though! If you place multiple elements with the same name inside another and you don’t provide identifiers, the first element that matches this will ALWAYS be used. The path: Switchers.Switcher Id=”1”.Inputs.Input.IsInstalled would always return a true (or Digital High) value, because it would always use the value of the element Input Id=”1”.