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

Elegant XML – Simple Serial Manager v1.2

GENERAL INFORMATION

SIMPL WINDOWS NAME	Elegant XML – Simple Serial Manager v1.2
CATEGORY	Elegant Modules
VERSION	1.2
SUMMARY	Provides a simplified module for reading nothing but Serial values out of an XML file.
GENERAL NOTES	This module works stand-alone to read/write Serial values from an XML file.
CRESTRON HARDWARE REQUIRED	3-Series Processor

PARAMETERS

FileName	S	The file name on the processor to load. By default if this is just a file name this will check for a file in the directory User/Config/App# (where # is equal to the program slot the program is running in). First it will check the ROMDISK/Config/App# path to see if the file exists there. If it doesn't exist in either of the two spaces, a default file is created in User/Config/App#. If you prefix the string with a \ you can provide your own full path. For instance: \\RM\\MyFiles\\Config.xml

RootElement	S	This is the RootElement of the XML file. For example, if you use the word Config, then the processor will expect a file that looks like:
		`<?xml version="1.0" encoding="UTF-8" ?>`
		`<Config>`
		`<DATA />`
		`</Config>`

ManagerID	S	The unique ID of this Manager instance. All modules that are related to this manager will use this ID. Only one manager in a program should have this ID.
PathDelimiter	S	A single character that defines the delimiter to use when separating path elements in the path parameters.
DefaultValueDelimiter	S	A single character that defines the delimiter to use when separating the path from the default value in the path parameters.

XmlPath[#] (1 – 100)	S	The path to the XML attribute to read from the XML file.

CONTROL

LoadFile	D	On the rising edge, this attempts to load the file. This shouldn't be called until all dependent modules have initialized.
SaveFile	D	On the rising edge, this attempts to save the file. This shouldn't be called until all dependent modules have initialized.
EnableDebug	D	Hold this value high to enable debug messages to print to the console.
InputValue[#] (1 – 100)	S	When this values changes it sends the updated value to the Simpl# program, which stores it in memory. This value is then reflected out the associated OutputValue[#] join.

FEEDBACK

IsLoading	D	Indicates if the module is in the process of loading data from the XML file.
LoadSuccessPulse	D	Pulses for 0.1 seconds when the file is loaded successfully.
LoadFailurePulse	D	Pulses for 0.1 seconds when the file fails to load.
IsSaveNeeded	D	Transitions to high when a value has been changed. Note that the Simpl# code only tracks if any value has changed at least once. If a value is changed and then returned to its previous value, this will remain high.
IsSaving	D	Indicates if the module is in the process of saving data to the XML file.
SaveSuccessPulse	D	Pulses for 0.1 seconds when the file saves successfully.
SaveFailurePulse	D	Pulses for 0.1 seconds when the file fails to save.
Progress	A	Provides a value of 0% (0d) to 100% (65535d) to track the progress of the current save or load operation.
ErrorMessage	S	Provides a readout of the error encountered while saving or loading. This is updated when the LoadFailurePulse or SaveFailurePulse outputs are pulsed.
OutputVaue[#] (1 – 100)	S	Outputs the value read from the XML file on the XmlPath provided. Also outputs the value from a change on the associated InputValue[#] join.

ADDITIONAL DATA

Revision History	v1.2 – Initial Release
Additional Details	This module is designed to allow loading string values from an XML file into Crestron Simpl Windows code. Although it has built-in string values, you can associate additional modules with this processor and their values will load. This module automatically loads the XML file/Default Values (if

no file or value within the file is found) after a delay of 3 seconds during which additional modules have the chance to register. Due to this delay, you don't need to perform a load after all the modules have registered.

Notes

- You do NOT need to place the RootElement on each 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 and no default value is defined in the path, 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.

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”.