Creating a custom WUI (Web User Interface) control

The term ‘control’ is used throughout this document and refers to an SVG file that is displayed in the WUI editor and can be dragged and dropped in the WUI editor. All of the provided library objects are referred to as WUI controls.

When creating a custom WUI control, the basic idea is to copy one of the existing controls and rename it, or create one from scratch in which case all of the rules for layers and control parameters must be followed. Keep in mind that an existing control already has the correct layering and parameters in place. You start by needing a new control with a new look that works like one of the provided ones, and go from there.

Custom SVG control files should be stored in this folder: “C:\Program Files\ICL\ScadaBuilder\WUI User Library”. The user folder is provided to keep the custom controls separate from the provided ones so that the custom ones don’t get deleted with a ScadaBuilder upgrade. When ScadaBuilder is installed, the controls in the WUI Library folder are deleted and new ones are put in.

WUI Control system

Each WUI control is defined in an SVG file. All of the WUI editor library controls are provided as SVG files that are copied to your hard drive as part of the ScadaWorks installation.

The SVG file can be opened several ways:

1. In an SVG graphic editor such as InkScape to edit as a graphic.
2. In an XML editor to edit as an XML file.
3. In a standard text editor such as Notepad to edit as an XML file.
4. Drag and dropped onto a web page to see what the graphic looks like.

When you add a control to a page in the WUI editor, the SVG file is automatically copied into your ScadaBuilder project directory and downloaded to the controller when a download is done. An HTML file is downloaded along with the SVG files, and the HTML file that sits on the controllers built in HTTP server is opened in a web browser that shows the SVG graphics and the associated data from the controller registers. Everything that is associated with the WUI system is located on the controller and is a standalone system.

Control Types

Every WUI control has a type number associated with it. The code that operates on the control looks at the type and runs code that is specific to that type.

Control Parameters

Each control has a set of parameters that are specific to how the control operates. The list of parameters are presented in the WUI designer as a way to configure the control.

Custom WUI Control Creation

Below is a list of all possible control parameters for the controls that are included with ScadaWorks, as well as what can be used in custom controls.

1	Reg
2	FltVal
3	Color
4	ColorList
5	IntVal
6	Range
7	Combo
8	IntWidth
9	DecWidth
10	Clamp
11	Text
12	TxtAttr
13	FontName
14	FontSize
15	FrmAttr
16	ReadOnly
17	Security
18	Dsbl
19	Invisible
20	Invalid

The parameters are located in the SVG files ICL metadata section and control what configuration options the control has when it is added to a page in the WUI editor.

Refer to the “WUI Control Reference” document for a list of all control types that are provided with a ScadaWorks installation and their respective configuration parameters. Here is an example of the type 101 parameters:

Type=101 Format 1
Reg
FrmAttr
ReadOnly
Security
Dsbl
Invisible
Invalid

The format is usually “1”, but is “2” in some controls to accommodate making a new control that works the same basic way but has extra parameters. An example of this are the selector switches, where the ones with hardcoded colors use the example shown above, and the ones with selectable colors have extra parameters for the color selection but work the same way (select between 2 or 3 switch positions).

Custom WUI Control Creation

This is the format 2 version that is used for the selection switches that can have the color configured:

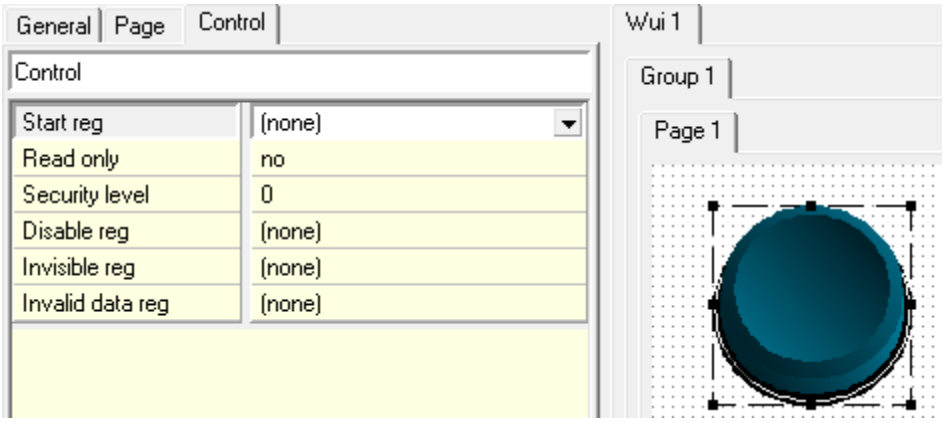
Type=101 Format 2
Reg
FltVal=Left Value,dflt=0
FltVal=Right Value,dflt=1
Color=Active Color
IntVal=Opacity %,dflt=50
FrmAttr
ReadOnly
Security
Dsbl
Invisible
Invalid

When copying and renaming a control to make a custom version, the format as well as the other parameters should not be changed. As mentioned previously, a specific type of control requires a specific set of parameters. Any custom control has to use one of the predefined types, and should include the original list of parameters for that type.

Take a look at the SVG / XML metadata section for the blue push button:

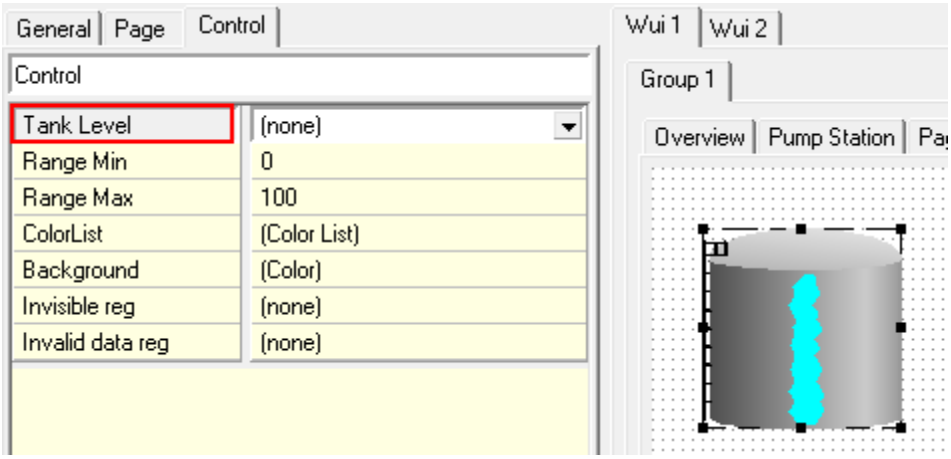
```
<metadata
  id="icl_wui">
 <prams
   id="prams79">
  Type=103
  Reg
  ReadOnly
  Security
  Dsbl
  Invisible
  Invalid
  help=Used to set a boolean value to a false/off (up) or true/on (down) condition.
 </prams>
```

Resulting configuration options in ScadaBuilder:



Renaming a parameter

A parameter can be modified so that a configuration name can be different when it shows up in ScadaBuilder. An example of this can be seen in any of the tank controls which show “Tank Level” in ScadaBuilder:



The XML metadata looks like

```
<metadata
  id="icl_wui">
 <prams
   id="prams55">
Type=102
Reg=Tank Level
Range=Range Min+Range Max,dflt=0+100
ColorList, Blink
Color=Background
Invisible
Invalid
help=Tank Dome Top - Has a level indicator with color control. The level color can change at specified levels.
</prams>
```

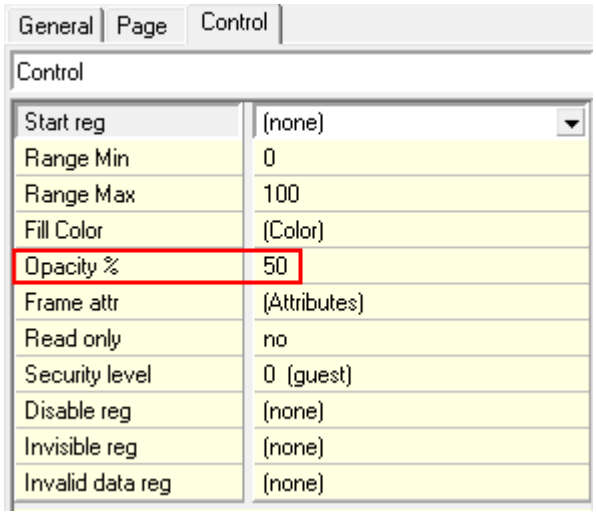
this:

Any parameter can be renamed by putting an “=” after the parameter, followed by the text you want displayed.

Setting a default value for a parameter

A control can be configured so that any of the parameters that use a numeric value (except the register parameter) can start with a default value. The tank XML above shows this being used to set the range parameters to 0 and 100. Another example of this is with the horizontal slider control. The XML and resulting WUI configuration look like this:

```
<metadata
  id="icl_wui">
  <prams
    id="prams7">
    Type=108
    Reg
    Range=Range Min+Range Max,dflt=0+100
    Color=Fill Color
    intVal=Opacity %,dflt=50
    FrmAttr
    ReadOnly
    Security
    Dsbl
    Invisible
    Invalid
```



A parameter can be set to a default value by putting 'dflt=' and the desired value after the parameter. Notice that in this case the parameter name and default values are both altered, and are separated by a comma. Double parameter modifications have to be separated by a comma (there are only two possible modifications, name and default value).

Working with Layers

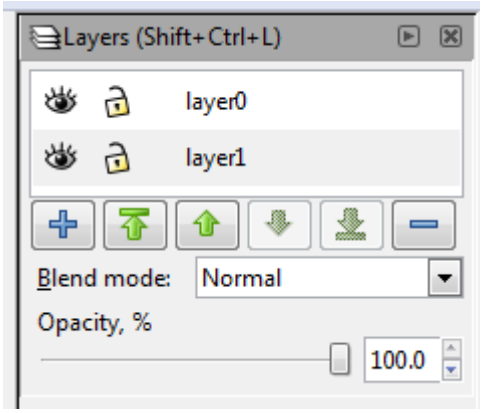
Each control type has a specific number of layers. The layers are viewed independently in InkScape. Animation for a control is accomplished by either moving, sizing, or rotating an object on a layer, or by turning layers on and off to display different 'frames' of animation. For example, the bargraph control changes the vertical size of the bar according to the register that is mapped to the control. A selector switch rotates the knob to one of 2 or 3 angle positions according to its mapped register value. And, a pump with a spinning animation uses frame by frame animation with each of the animated frames being on a separate layer. Each layer is turned on one at a time to give the appearance of a spinning wheel, and when the last frame has been shown, the sequence starts over.

There are some controls with the same type number that have different numbers of layers such as the 4 way, 3 way, and 2 way valves. The extra layers for the valves with less number of positions were deleted since they would never be made visible. This type of copying and editing can break a controls animation if not done correctly, so just be sure to test well with custom controls. Note that this differs from using parameters, where the list of parameters should always remain the same as the original.

This example uses the blue push button t see how layers work for animation.

Go to the "C:\Program Files\ICL\ScadaBuilder\WUI LIBRARY\Buttons and Switches\Buttons" folder and open the 'Push Button Blue' SVG in InkScape (or whatever SVG editor you are using). You can make a copy if you wish and open that one instead. Then go to the Layer menu and select "Layers" at the bottom. This button emulates a button that is on when pushed down and off when in the up position.

You'll get a Layer control area on the right side that shows the existing layers.



Test the animation by clicking on the eye icon for layer 0. If layer 0 is on then it is visible because it is on top of layer 1, giving the appearance of being up, released, or off. If layer 0 is turned off then layer 1 becomes visible which gives the appearance of being pushed down, or on. The control works on the web page by turning layer 0 on and off according to the value of the register that is mapped to the control (see the table below). Note that any of 3 numerical register types (Boolean, Integer, or Real) can be used.

	Button Condition	
	OFF	ON
Boolean	FALSE	TRUE
Integer	0	non-zero
Real	0.0	non-zero

Using layers has some rules that must be followed:

1. All objects and pieces that form objects (a rectangle, polygon, text, etc.) must be on a layer to be visible on the web page. This is mentioned because it is possible in InkScape to place an object outside of all layers.
2. An SVG file must have at least one layer which should be designated as 'layer0'. Naming layers will be discussed next on in this document.

Naming Layers

When looking at the layer control panel in InkScape, you'll notice that adding a layer has to be done above or below the currently selected layer. This is because SVG layers work in a top down fashion and only the top most layer, that is enabled/turned, on can be seen. You can see this happening by turning the top layer on and off, or by leaving the top layer enabled and turning the one below it on and off. Also, If you have a lower level layer enabled, select any object on that layer then disable that layer, the selected object remains selected even though you can't see it anymore. This is important to remember because you can accidently delete an object without realizing it.

The ICL method for naming layers is necessary to follow because the code that controls layer based animation needs to have the layers named a specific way. Our method requires that you name either the layer at the top or at the bottom of all other layers as 'layer0'. The other layers going up or down from that layer are named in succession as 'layer1', 'layer2, etc. Typically layer 0 is referred to as the static layer, and all others are used to control the animation. Remember that a layer that is enable and is on top of another layer will be the only one visible (In InkScape and on the web page), and the name of the layer has nothing to do with where the layer is located in the top down architecture. The code just uses the layer name to determine which layer to enable to get the desired viewing effect. As mentioned before, if an SVG has only one layer, the layer should be named as 'layer0'.

InkScape issues

Transforms – Sometimes when moving things around in InkScape the movement is remembered by InkScape as a transform. Transforms are ok everywhere in the SVG file except in a layer definition. You check for this by opening the file as XML and looking in the ICL metadata section. If a transform shows up in a layer section then it has to be removed. Be careful when doing this and don't take anything else out or the file will break and cannot be opened anymore in Inkscape.

This example has a transform in layer 1:

```
<g
  inkscape:label="Layer 1"
  inkscape:groupmode="layer"
  id="layer1"
  transform="translate(0,-952.36217)">
 <rect
    style="fill:#666666;fill-opacity:1;fill-rule:nonzero;stroke:none"
    id="rect2985"
    width="71.428574"
    height="82.857147"
    x="17.142857"
    y="960.93359" />
</g>
</svg>
```

Here the transform is removed:

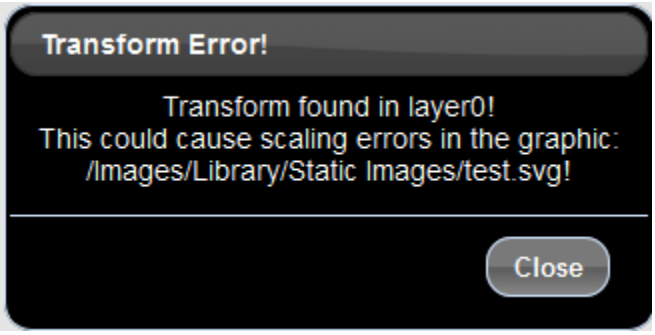
```
<g
   inkscape:label="Layer 1"
   inkscape:groupmode="layer"
   id="layer1">
  <rect
     style="fill:#666666;fill-opacity:1;fill-rule:nonzero;stroke:none"
     id="rect2985"
     width="71.428574"
     height="82.857147"
     x="17.142857"
     y="960.93359" />
 </g>
</svg>
```

Note that the' >' that was after the transform **was not** removed which is what we want.

Now when the file is re-opened in InkScape, the moved object will be somewhere outside of the document box. Sometimes you need to zoom way out to find it:

Move any objects back into the document box to fix the issue:

The WUI will let you know if there is a transform in any layer section of the XML metadata when you open the web page:



Remember to check all layer sections in the XML where the ICL metadata resides.

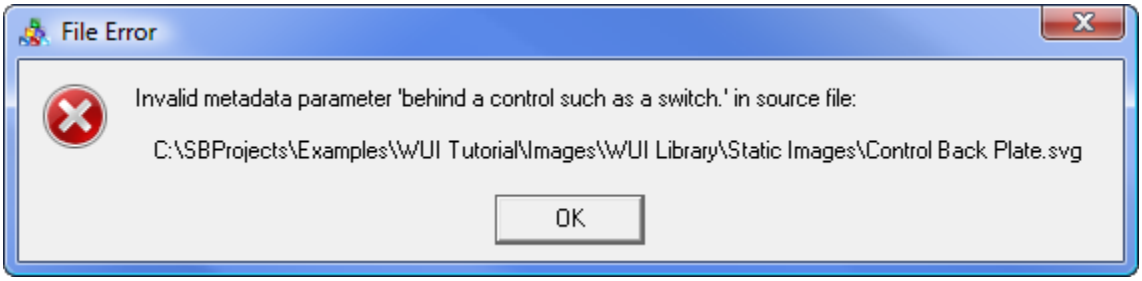
Quotes and commas – In the ICL metadata section of the file XML you can put in a help tip that shows up at the bottom of ScadaBuilder when the control is dropped onto the designer.

```
<metadata
  id="icl_wui">
 <prams
   id="prams14">
 Type=100
 Invisible
 FrmAttr
 help=Control Back Plate - Gives the appearance of a metal plate behind a control such as a switch.
```

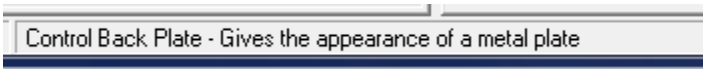
If you wish to have any comma's as part of the help message, you have to put the entire string in double quotes:

```
help="Control Back Plate - Gives the appearance of a metal plate, behind a control such as a switch".
```

Without the double quotes, you'll get an error:



And the message will get cut off at the point of the comma:



With the double quotes, the control will work just fine for ScadaBuilder, but if you open the file in InkScape and change anything about the file, the double quotes get replaced by &quot:

```
help=&quot;Control Back Plate - Gives the appearance of a metal plate behind a control such as a switch.&quot;
```

The help message will then show like this in ScadaBuilder. To fix this go back into the XML and fix the help message.