Opendiem Training

Exercise 7

Opendiem-TRN-0007

Revision	5.0.0		
Status	Initials	Date	Comment
Released	RAC	11/18/2010	Updated format and content

Exercise 7 – Advanced Opendiem SmartComponents

Introduction

This exercise is an introduction to creating Opendiem SmartComponents. SmartComponents are different from other Opendiem shapes in the sense that they are not shapes at all. A SmartComponent is a pre-defined object which can have intelligent associations with the data and is a powerful feature of Opendiem that compliments the shapes and allows complex screens to be quickly and easily designed.

SmartComponents are linked rather than added to the screen and loaded dynamically when the screen is being viewed. If changes are made to a SmartComponent on the disk then these changes will be reflected on all screens that use that object.
The power behind the SmartComponent is the ability to give it intelligence. A SmartComponent can be associated either with a single variable or a set of variables, such as different variables on a single controller, even different variables on multiple controllers of a complete sub system.

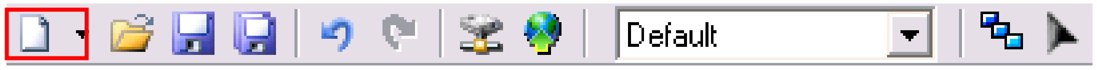
Objective

In this exercise you will create advanced Opendiem SmartComponents with Tags to control animation and color states.

Re-opening the project in Opendiem Designer

Exercise Instructions

Ensure that Opendiem Manager is running on your Opendiem Server. Restart if necessary.
If needed, re-open the previous project using Opendiem Designer.
We will open a new screen in which to create our SmartComponent.
Create a new screen by clicking on the **New** Item button on the toolbar.

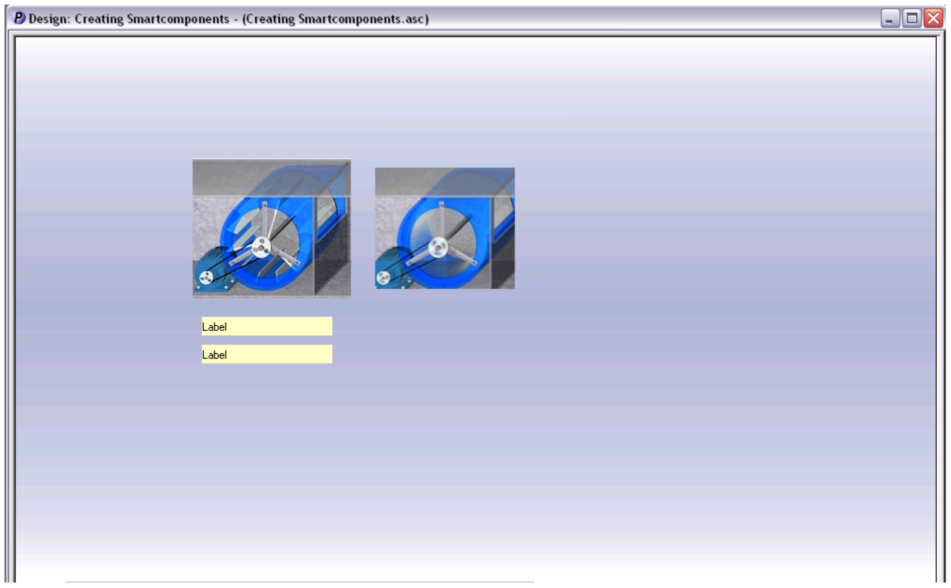


In the Screen Properties dialogue box which appears enter a screen title, e.g. 'Creating Smartcomponents', and click **OK**.

The first step in creating a smartcomponent is to drag in the image or object on which the SmartComponent is going to be based. Let's make an AHU fan into a smartcomponent. Locate the Graphic called: `AHUFanCroppedRightExpelLarge.gif`, and drag it onto the screen.
Click on original size so that you don't have to resize it. This is the main fan graphic around which we will build the SC. As this is going to be an animated SC we also need the animated version of the image we just dragged on.
Now locate and drag onto the screen: `AHUFanCroppedRightExpelAnimated.gif`
Make sure that original size is selected and that you have specified the correct number of frames in the file (in this case 3).
Now that we have the main components on the screen lets drag some labels onto the screen to represent some data.

Drag a label onto the screen and change the color option of the label to light yellow (the first yellow on the color palette)

Duplicate this label so that you have two labels, place the duplicate below the first. Your screen should look like the one below:


Design: Creating Smartcomponents - (Creating Smartcomponents.asc)
Label
Label

We are going to represent the alarm state on one label and the fan state on the other.
We want to make it editable so that once it has been made into a SmartComponent the user can drag it onto the screen and assign any values he or she would like the SmartComponent to have.

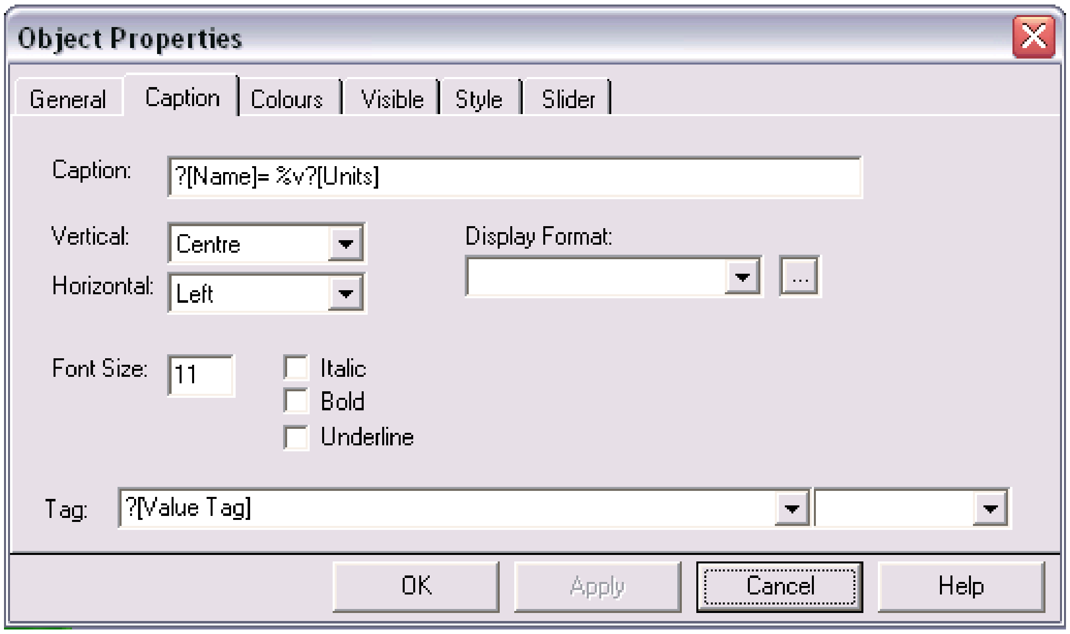
For the first label, we want it to have fields for Name, Value and Units.
We also want all these values to be editable.
Click on properties of the first label and go to captions. On Caption, where is says Label, replace Label with the following:

```
?[Name]= %v?[Units]
```

The '?' represents the fact that what is to be displayed within the brackets is to be input by the user. Although we have written 'Name' within the brackets, this will not appear in the output and is merely the description of the placeholder. Next we put equals and a space so that the data does not appear immediately after the title. The '%v' is the placeholder for the actual value to be displayed. The value is pulled from the Value Tag at the bottom of object properties. Finally we have a placeholder for the user to input the required units.
We are now going to put a placeholder in the Value Tag. Go to Tag and write:

```
?[Value Tag]
```

This will enable the user to input the value they want displayed on the label. Now we can make a few cosmetic adjustments. Set font size to 11 and make the horizontal and vertical alignment equal 'center'.



Caption: ?[Name]= %v?[Units]
Vertical: Centre
Horizontal: Left
Font Size: 11
Italic [ ]
Bold [ ]
Underline [ ]
Tag: ?[Value Tag]

We have now completed the first label.

We want the second label to show the name and two alarm states 'Normal' and 'Fault'. Click on the second label's properties, and go to caption. In the caption filed type:

```
?[Alarm Name]= %v
```

The naming follows the same principles as that of the first label. In the Value tag type:

```
?[Alarm Tag]
```

Now we will set the two alarm state which we want to appear.
Click on the more options button next to the 'Display Format:'. You will see a new popup panel called: 'Caption Format'.
Select Text Label and click Next. Select User Defined and click Next. You will now see two columns, one containing values and the other containing fields for labels.
As the Alarm tag will be working off a Boolean operator, we only need the values 0 and 1. We want the label 'Normal' to appear on 0 and 'Fault' to appear on 1. So next to 0 and 1, enter Normal and Fault, respectively. Click, Finish.

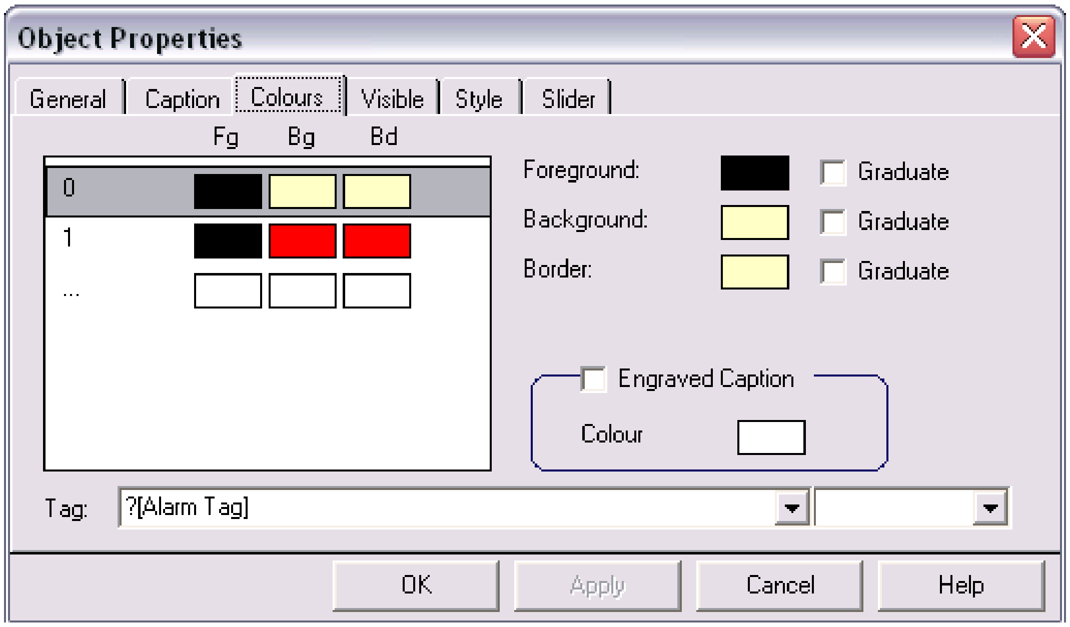
Now we have the Name, Value Tag and the labels which we want displayed. Now, as before, set font to 11 and make the alignment equal center.

We want to add an extra functionality to make the text label change color to red when the alarm state is fault. Go to the color option. We want to set the color to turn red when the Boolean operator is 1 (When the alarm state is Fault).
The color is set, so that on 0, the text is black and the background is the yellow color we chose earlier. Below the first color value, there are provisions to put in more color states.

Enter 1 into the threshold value below the first and select black as the text color and dark red as the background color. Now we have to put in the Tag which will switch the color state from the first to the second. As this is going to be the same as the value which controls the labels, in the Tag field, enter:

```
?[Alarm Tag]
```

The object properties box should now look like this:



	Fg	Bg	Bd
0	⬛	🟨	🟨
1	⬛	🟥	🟥
...	⬜	⬜	⬜

Foreground: ⬛ Graduate ☐
Background: 🟨 Graduate ☐
Border: 🟨 Graduate ☐

Tag: ?[Alarm Tag]

Now we have the two labels we wanted.

Next position the animated fan image over the still fan image so that they are properly aligned. We want the animated image to only become visible and animate if the fan state is greater than 0 (i.e. if the fan is on). How do we do this?
Click on the animated fan object properties and go to Visible options. In the Tag write:

```
?[Value Tag]
```

This is the same name as that of the placeholder in the fan state label, as they are both representing and being triggered by the same value. Also above the Tag, from the five options, select 'Greater Than' and make sure the Animate Value is 0.
Now the fan animation will only be visible when the Value from the input is greater than 0.

We are almost ready to make the smartcomponent.

But let's make it a little more advanced so that labels only appear when the user passes their mouse over the fan image.
Drag a panel onto the screen big enough to cover the whole area of the fan.
Click on color options and select 'No Fill' on all the boxes in the color palette, so that the panel appears invisible.
Drag the two labels over the fan and animated fan images. Find what the names of the labels are. This can be done by clicking on the object property box of the labels and clicking on General and reading the Item Identifier.
We want these labels to appear when the user passes his mouse over the invisible panel which will be placed over the fan. On the panel click on actions and set the actions so that on mouse over the labels are visible and invisible on mouse exit.

As we don't want the labels to appear Onload, but to be triggered by the mouseover action, click on the visible properties of the labels and uncheck the visible property.
Click apply and click OK. Now position the panel over the fan images and the labels.

Now Select all and use the SmartComponent Wizard to convert it into a SmartComponent called BlueFan.

The next stage in creating the SmartComponent is to create the associations between data types and the SmartComponent files.
These associations are held in a file called Associations.cfg.

1. Open the 'Associations.cfg' file in notepad. The file is located in the Opendiem Library folder located at
   'c:\Program Files\Building Clouds\Opendiem\Library'

2. Enter the following text:
   [number]

```
0=BlueFan.csc
Value Tag=[node]
Title=Blue Fan AHU
```

3. When you dragged the nvoRoomTemp network variable onto the design screen in a previous exercise, Opendiem presented a list of SmartComponents for selection. This list of SmartComponents is read from the Associations.cfg file and the first line of each section defines the data type or controller type that the SmartComponent relates to. It is possible to associate a SmartComponent with more than one data type by adding multiple entries each with a different data type header.

   The second line contains the SmartComponent file that describes the SmartComponent for this association and always takes the form 0=filename.csc.

4. Subsequent lines in the section assign values to the Property Markers defined as part of the SmartComponent .csc file. In the example above you will note the special key '[node]' this is a reserved word in the Associations.cfg file and is replaced with the variable name during a drag & drop operation:

   For example: If the Lonworks NV LON.Subsystem 1.node.nvoDigital is dragged and dropped onto the design screen then the following substitution may occur:
   ```
   [node].state  -> LON.Subsystem 1.node.nvoDigital.state
   ```

   When a variable of type Blue Fan is dragged onto the design screen this instructs Opendiem to present the Blue Fan.csc SmartComponent as an option in the pop-up list.
   If this SmartComponent is selected the line Tagname=[node] informs Opendiem to substitute the Property Marker 'Tagname' with the absolute NV reference dragged onto the design screen e.g.
   ```
   LNS.SUBSYSTEM 1.FCX.NVISETPOINT
   ```

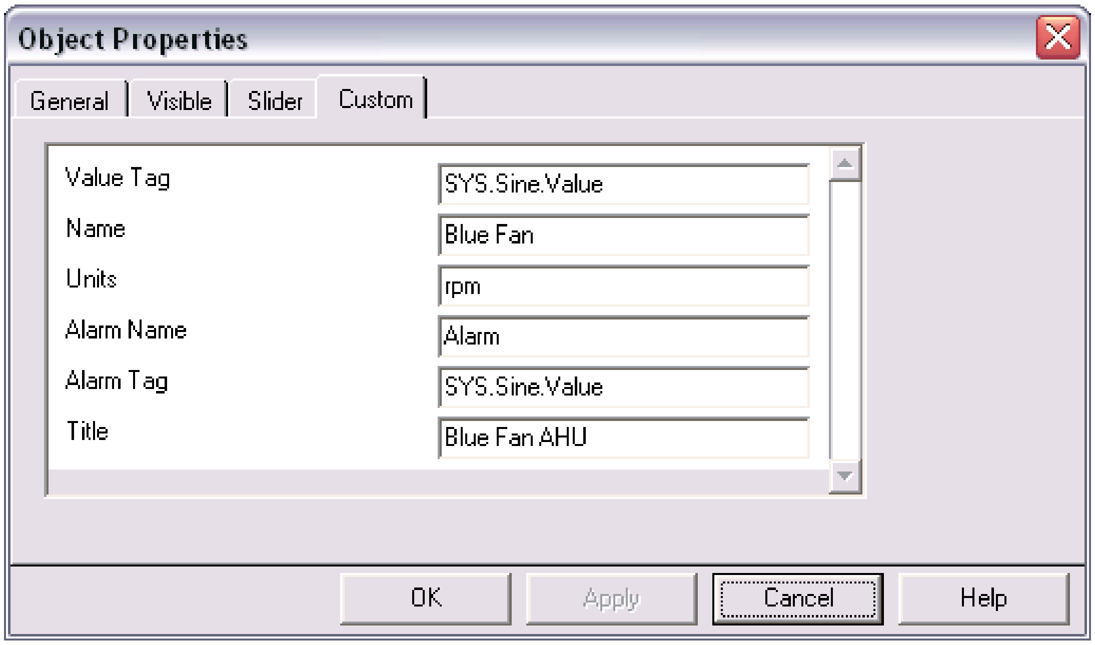
5. The section 'Title=Blue Fan' will provide a SmartComponent title attribute that can be edited by the user of the component.

6. The association is now complete, save the file.

7. To access your new SmartComponent, expand Drivers, Sys, Ramp in the Project Explorer window and drag Value onto the Screen. You should now have a list of options in which you should be able to see 'Blue Fan'. Note that the name in the pop-up list is the screen title of the design page that you originally created the SmartComponent .oscreen file as.

8. Select the Blue Fan SmartComponent and it will appear in your design screen.

9. From Designer right click on the SmartComponent and select **Properties** you will see a box of editable options as below. This has the different placeholder which we entered when we created the SmartComponent as well as the Title below. The values can be edited as desired.



Value Tag	SYS.Sine.Value
Name	Blue Fan
Units	rpm
Alarm Name	Alarm
Alarm Tag	SYS.Sine.Value
Title	Blue Fan AHU

End of Exercise 7

In this exercise you have created an advanced SmartComponent that allows you to add pre-built components to an Opendiem design that has automatic associations with data types. You have reviewed the way that Property Markers are defined and how the values are substituted by the Associations.cfg file.

This exercise has not fully explored some of the advanced features of SmartComponents, which are beyond the scope of this training class.

Notes:

Version 5.00 November 2010

Building Clouds
3229 Whipple Road
Union City, CA 94587

Email: support@buildingclouds.com
http://www.buildingclouds.com