Adroit Smart SCADA

Object Modelling Quick Start

Contents

Introduction

For high-end System Integrators and OEMs, the holy grail of SCADA has been the dream of creating your own fully-fledged, custom, industry-specific SCADA database types along with graphical user interface representations to visualize and interact with them. Because, at its heart, Adroit Smart SCADA is an extensible object model, we show how this dream is now a reality.

In this *Object Modelling* quick start guide, we create a custom SCADA database object type complete with its own set of *properties* and scripted *methods* in keeping with a conventional object oriented paradigm. After verifying its successful operation, we go on to develop a re-usable graphical representation that can be deployed to visualize instances of the object type. Finally, because alarming and alarm management are so vital to SCADA applications, we demonstrate firstly how to create custom alarm types, and then how to smoothly integrate these with the Adroit alarming subsystem.

This is the fifth in the series of Adroit Smart SCADA quick start guides. It follows on from Quick Start Guide 1 – the initial Adroit Smart SCADA Quick Start Guide, Quick Start Guide 2 – Logging & Trending, Quick Start Guide 3 – Alarming & Alarm Management, and Quick Start Guide 4 – Smart UI Re-use Techniques. There are some aspects in each of these earlier guides that are essential to understanding this latest guide, and so in that respect they are all, to some extent, essential prerequisites.

Note: You may prefer to follow this quick start online at http://adroit-europe.com/om as the screenshots render more precisely, and the content is updated from time to time.

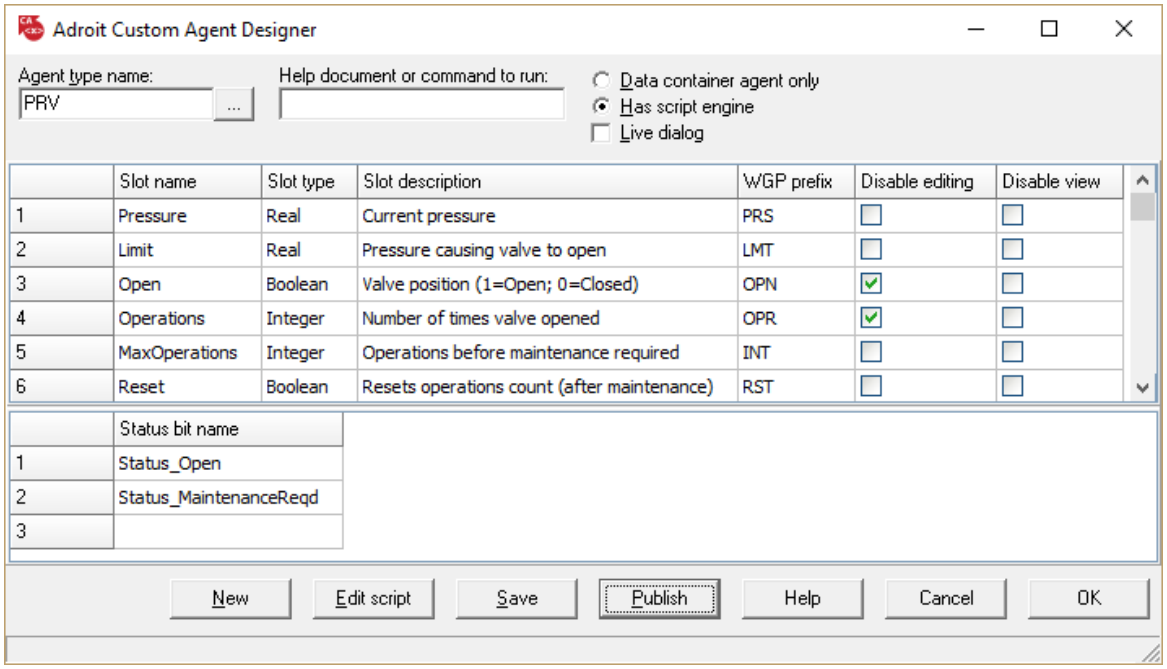
PRV Custom Agent Type

To keep it all relatively understandable and yet fairly meaningful, we create a hypothetical agent type that models a Pressure Relief Valve (PRV). The characteristics of our PRV are that for pressures higher than some configurable limit the valve opens, but for lower pressures it remains closed. In addition, a count of every open operation is maintained so that essential maintenance can be indicated after a configurable number of operations. Our valve can raise two kinds of alarms: one when it is open, i.e. excess pressure, and a second when essential maintenance is due.

As shown in the screenshot of the Adroit 8.4 program group, a special tool for developing custom agent types ships with Adroit: *Custom Agent Designer.*

This tool has its own, comprehensive, in-built Help repository, so run the tool, click on the *Help* button, and familiarize yourself to some level of detail with its features and operations before continuing any further.

The screenshot below shows the design for our Pressure Relief Valve custom agent type:



- Top left is the name given to this agent type – PRV for Pressure Relief Valve

- In the central area are definitions for 6 slots (properties) a PRV agent has:

Pressure	A Real number representing current pressure. The *WGP Prefix* column is a key or prefix used when saving the value of this slot away in the agent server database
Limit	Another Real number defining the limiting pressure above which the valve will operate, i.e. open. Because this value is a separate property it means that different PRV instances can have different operating pressures
Open	A Boolean value indicating whether the valve is open or not. The value of this slot will be determined purely by whether the current pressure is above or below the limiting pressure. Hence the *Disable editing* checkbox is ticked so as to prevent users and other entities external to the agent itself from modifying the value of this slot
Operations	An Integer number representing the number of times the valve has operated since essential maintenance has been carried out. The value of this slot is also controlled purely by the agent itself, hence *Disable editing* checkbox is ticked
MaxOperations	Another Integer number defining the number of operations that may be carried out before essential maintenance or servicing is required. Because this value is a separate property it means that different PRV instances can have different maintenance regimes
Reset	A Boolean value used to reset the operations count after essential servicing has been done

- At the bottom are two type-specific status bits a PRV agent has:

Status_Open	ON when the valve is open; OFF otherwise
Status_MaintenanceReqd	ON when maintenance is due; OFF otherwise

Because some of the slots such as *Open,* and *Operations* as well as the two type-specific status bits need to be controlled by the agent itself in response to value changes in various other slots, it is necessary to make sure each agent has a *script engine* for this purpose. This is indicated in Custom Agent Designer by selecting the *Has script engine* radio button top right of the above screenshot.

Our *VBScript* implementing the PRV agent logic is shown below. The screenshot shows the VBS script file loaded into Visual Studio – hence the nice colour-coding. Just clicking the *Edit script* button in Custom Agent Designer will, however, invoke a much simpler script editor that does not have the same colour-coding

```
Option Explicit 'force variable declarations to prevent compile errors

REM Slot indices
Const idx_Pressure = 0
Const idx_Limit = 1
Const idx_Open = 2
Const idx_Operations = 3
Const idx_MaxOperations = 4
Const idx_Reset = 5

REM Define the status bit index numbers starting at 16
Const idx_Status_Open = 16
Const idx_Status_MaintenanceReqd = 17

Sub OnCompiled()
  'global initialisation - called for each agent at startup and whenever this file is saved while AS is running
  Custom.LogEvent "Compiled OK"
End Sub

Sub OnChange(slotid, value)
  'agent slot changed - called whenever a putslot has been done into one of this agents slots

  Select case slotid
    'Pressure slot has changed
    Case idx_Pressure
        Custom.LogEvent "Pressure slot change: " & value
        if value - Custom.GetSlot(idx_Limit) > 0 Then
            'Pressure is above limit
            Custom.LogEvent "Above Limit"
            if Not Custom.GetSlot(idx_Open) Then
                'Valve must be open
                Custom.SetSlot idx_Open, 1
                Custom.SetStatus idx_Status_Open, 1
                Dim newCount
                newCount = Custom.GetSlot(idx_Operations) + 1
                Custom.SetSlot idx_Operations, newCount
                if newCount - Custom.GetSlot(idx_MaxOperations) >= 0 Then
                    'Maintenance is required
                    Custom.LogEvent "Maintenance Required"
                    Custom.SetStatus idx_Status_MaintenanceReqd, 1
                End If
            End If
        Else
            'Pressure is within limit
            Custom.LogEvent "Within Limit"
            if Custom.GetSlot(idx_Open) Then
                Custom.SetSlot idx_Open, 0
                Custom.SetStatus idx_Status_Open, 0
            End If
        End If

    'Reset slot has changed
    Case idx_Reset
        if value Then
            'Maintenance has been carried out
            Custom.LogEvent "Maintenance Done"
            Custom.SetSlot idx_Operations, 0                    'Reset operations count
            Custom.SetStatus idx_Status_MaintenanceReqd, 0      'Clear maintenance required status
            Custom.SetSlot idx_Reset, 0                         'Reset this slot (idx_Reset)
        End If

  End Select
End Sub
```

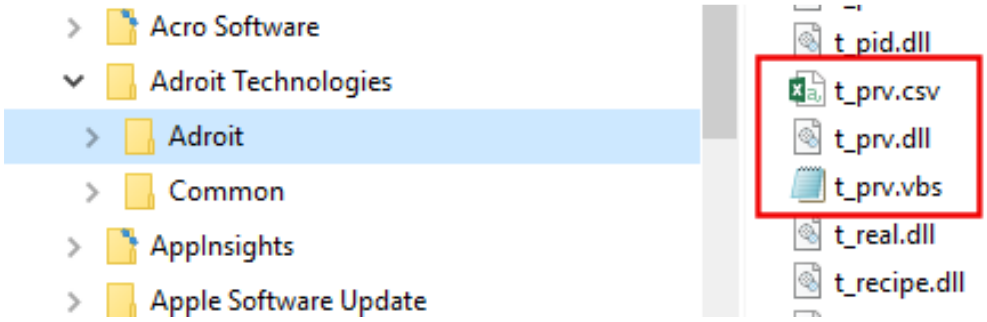
Understanding the PRV agent script

Starting at the top of the above script:

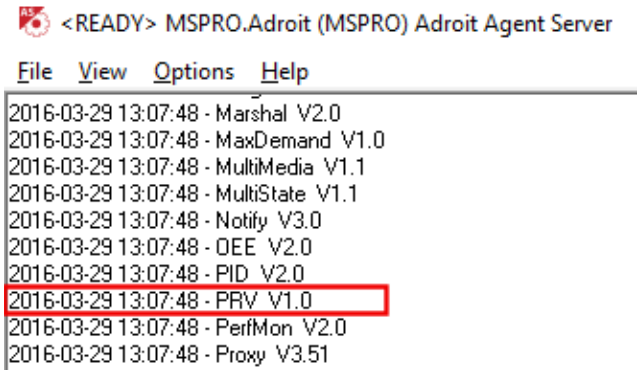
- *Option Explicit* is a Visual Basic statement which forces variables to be declared before they can be used and is regarded as a way of reducing the likelihood of producing code that will generate compile errors
- As described in the Custom Agent Designer Help, it is necessary to assign unique values starting from zero and incrementing for each different slot defined in the agent. So, by means a *Const* statements, we create six different constants – one for each slot
- Similarly, we define two different constants – one for each type-specific status bit. Type-specific status bits need to be given indices starting from 16 up to a maximum of 31. This is because status bits 0 through 15 have predefined functions common to all agent types, whereas status bits 16 through 31 are free to be defined for any purpose the design of an agent type needs
- A handful of *Script Procedures* are described in the Help. To implement the PRV agent we need only *override* two of these: *OnCompiled* and *OnChange*
- OnCompiled is really there only so that compilation errors can easily be detected. All we do is call the *Custom.LogEvent* script method (cf. Help) and write the string "Compiled OK" to the Windows event log. When first developing the script if you see this message in an event view, then you can be sure the script has compiled OK. Otherwise there should be some other message in the event log describing any compilation error(s)
- OnChange is the real workhorse procedure of the PRV agent and is called whenever one of its slot values is modified, for example whenever the valve pressure changes. The only two slots we need to be concerned about in the PRV script engine are the *Pressure* slot and the *Reset* slot. So we code a *Select case* statement with cases *idx_Pressure* and *idx_Reset*
- In the case of the *Pressure* slot being changed
    - For purely diagnostic purposes, we log an event that indicates this and includes the changed value of the slot
    - Next, using the *Custom.GetSlot* method, we test if the changed *Pressure* slot value is greater than the *Limit* slot value
        - If so, pressure is above the operating limit, so we log this fact, and if the valve is not already open, using *Custom.SetSlot* method calls we set both the *Open* slot and the *Status_Open* status bit. The reason we need to set a status bit as well as a slot is so that an *Open* alarm can be raised when this happens
        - Next, via *Custom.GetSlot* and *Custom.SetSlot* operations on the *Operations* slot, we increment the *Operations* slot. If the new count of operations has reached the maximum allowed before needing maintenance, then we log this fact and set *Status_MaintenanceReqd* status bit so that a *Service Due* alarm can be raised
    - If the changed pressure value is within limit
        - We log this fact
        - If the valve was open, we set it closed by setting the *Open* slot value to zero, and also setting the *Status_Open* status bit to zero, and thereby clearing the *Open* alarm
- In the case of the *Reset* slot being changed to a value of *On*
    - This implies maintenance has been carried out, so for diagnostic purposes we log this fact
    - We set the *Operations, Status_MaintenanceReqd,* and *Reset* slots to zero

Running the PRV Custom Agent

Before being able to try out our new custom agent type, we need to publish it by clicking the *Publish* button on the Custom Agent Designer screen, which will have the effect of copying all T_PRV.* files into the Adroit installation folder as shown

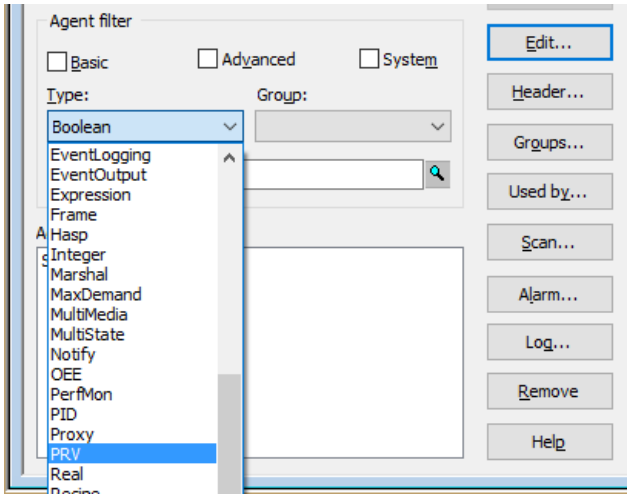


The CSV file contains the various slot and status bit definitions, etc. The VBS file contains the Visual Basic Script described above, and the DLL file is the actual custom agent DLL loaded into the agent server at start up. A further message box is displayed, explaining that these custom agent files will need to be manually copied to any other network computers that are part of the solution.

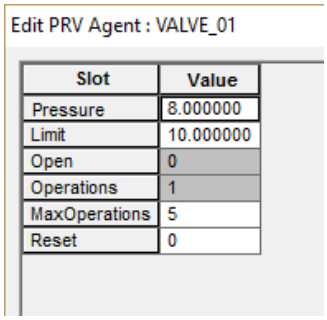


At this point we can run up the agent server application with the same configuration that was running at the end of the most recent Quick Start Guide 4 on Smart UI Re-use.

As shown alongside, our new PRV custom agent type will be shown along with all the other built-in Adroit agent types.



The PRV agent type will also be visible as a *Type* along with other agent types. PRV *agent instances* can be created, edited, and otherwise configured.

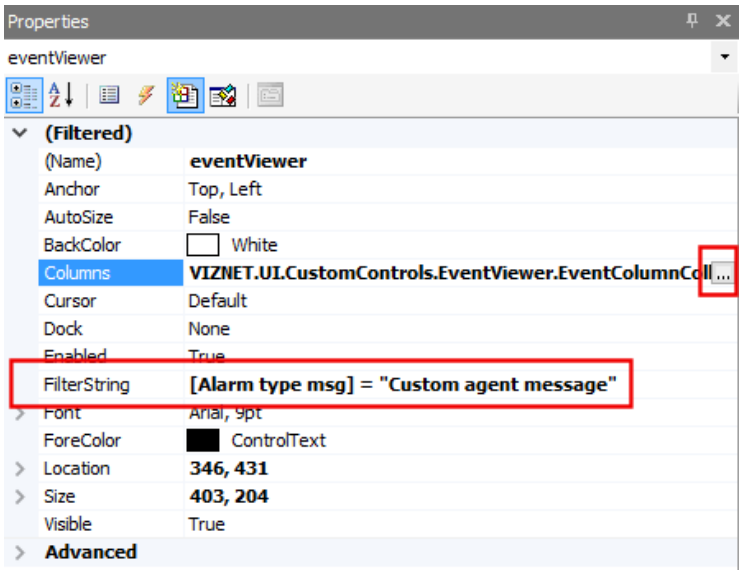


The right-hand screenshot shows we have created PRV agent VALVE_01 and configured *Pressure*, *Limit*, *MaxOperations*, and *Reset* slots to be 8.0 psi, 10.0 psi, 5, and 0 respectively. Note that because *Open* and *Operations* slots are not editable, these are shown in grey background

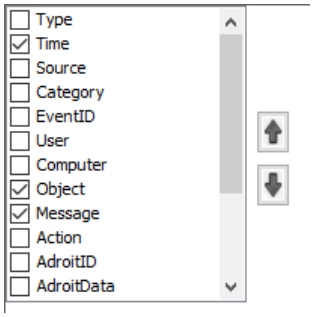
Testing and diagnosing the PRV Custom Agent

The reason we have included so many *Custom.LogEvent* method calls in the PRV agent script engine is so that we can relatively easily see what's going on at run time. When you are satisfied everything works as expected, you can take these statements out of the script if you want to.

In the meantime, create a *Graphic Form* and drop an Adroit *EventViewer* control onto the graphic form



To avoid getting all sorts of Adroit events displayed, set the *FilterString* property as shown. This will mean only event messages originating in a custom agent will be displayed



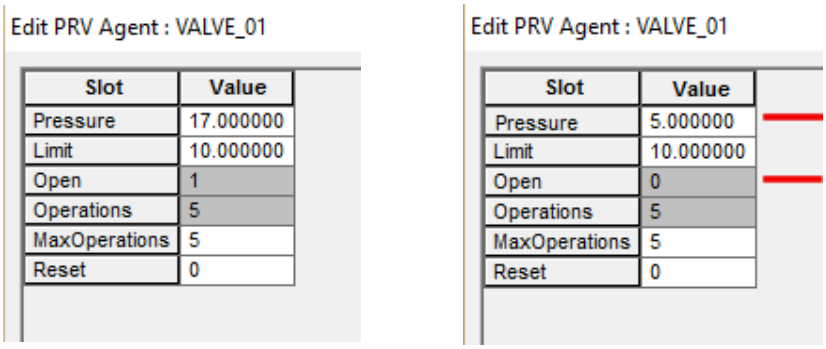
Next, use the *Ellipsis* […] button at the right of the *Columns* property to select and order a sub-set of columns to be displayed. All we really need to see are the *Time, Object,* and *Message* columns, after which the event viewer should look something like the screenshot below when the form is run either in Smart UI Designer or Operator

Time	Object	Message
2016-03-29 13:49:14	VALVE_01	Maintenance Required
2016-03-29 13:49:14	VALVE_01	Above Limit
2016-03-29 13:49:14	VALVE_01	Pressure slot change: 17.000000
2016-03-29 13:49:08	VALVE_01	Within Limit
2016-03-29 13:49:08	VALVE_01	Pressure slot change: 7.000000
2016-03-29 13:49:04	VALVE_01	Above Limit
2016-03-29 13:49:04	VALVE_01	Pressure slot change: 17.000000

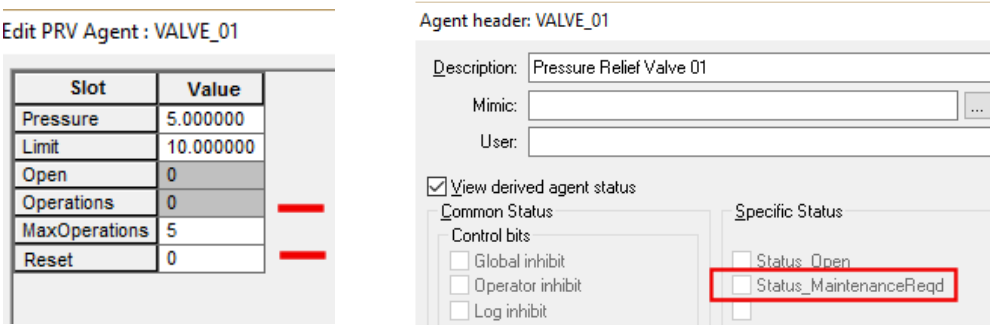
From the *Time* column we can see that events are ordered in reverse chronological order. The earliest event we see is a pressure value change to 17 psi which is above limit and should cause the valve to open. Next we see a pressure value change to 7 psi which, being less than 10 psi, is within limit and should cause the valve to close. Finally we see another, above limit pressure value change, and in this case because the total number of open operations has exceeded 5, a *Service* alarm is raised as indicated by the *Maintenance Required* event message. Note: there are only two open operations visible on the event viewer but if you *paged* up and down using the *Page Up* and *Page Down* buttons on the event viewer you would see earlier open operations.

All this will be a lot clearer when we develop and show different PRV instances using a Smart UI wizard and alarm view, but for now we need to be sure the custom agent is functioning correctly and to test this we can simply modify slot values in the Configurator and observe the effect on other slots.

The screenshot below left shows the current situation, i.e. valve open due to pressure above limit at 17 psi. The right-hand screenshot shows when we bring the pressure back down to within limit (5 psi), the valve closes again (Open = 0)



Next if we set the *Reset* slot *On* three things happen: The *Status_MaintenanceReqd* bit in the agent header is reset (right-hand screenshot below), the *Operations* slot is reset to zero, and the *Reset* slot itself reverts to *Off*



Continue experimenting for yourself with various slot value changes until you are happy from observing the event view and other slots that everything is as it should be.

Visualizing the PRV Custom Agent

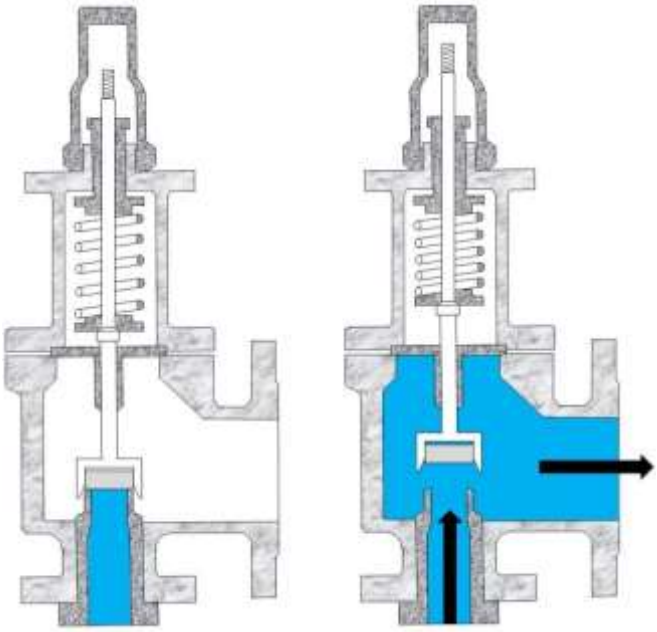
Our goal in this section is to create a comprehensive Smart UI wizard that can be used to visualize all salient aspects of PRV agent instances succinctly. As we learned in the previous quick start guide on Smart UI Re-use techniques, the best way to go about creating a wizard is first to create a graphic form for a single agent or known group of agents and then later on, turn this into a substitutable wizard.

Accordingly, the following table lists the type of graphical elements we would like when visualizing PRV agent instances on a graphic form, after which we will drill down in some detail on how each different element is configured.

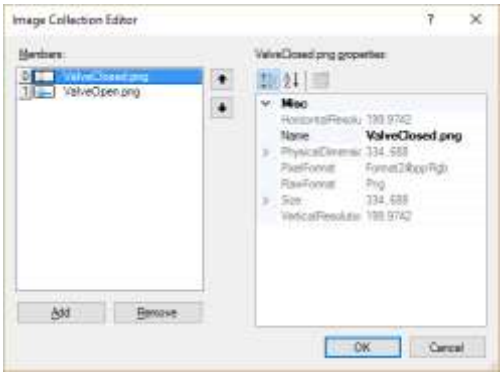
Type of element	Description of use
Picture	• Picture element to (a) show valve in two different positions: Closed and Open and (b) display a tool-tip when the mouse hovers over the picture
Text	• Text element to display current pressure value • Text element to display a description for PRV agent instance • Text element to display text *Open* when valve is open • Text element to display text *Service Due* when maintenance is required • Text element to display alarm priority of *Open* alarm • Text element to display alarm priority of *Service* alarm
Ellipse	• In line with high-performance HMI situation awareness guidelines (ISA 101), a circle element in Orange to indicate *Normal* priority *Open* alarm
Polyline	• Similarly, a triangle element in Yellow to indicate *High* priority *Service* alarm
Button	• Button element to set *Reset* slot *On* after maintenance has been carried out
Trackbar	• A slider element to simulate the process by varying the pressure up and down as required

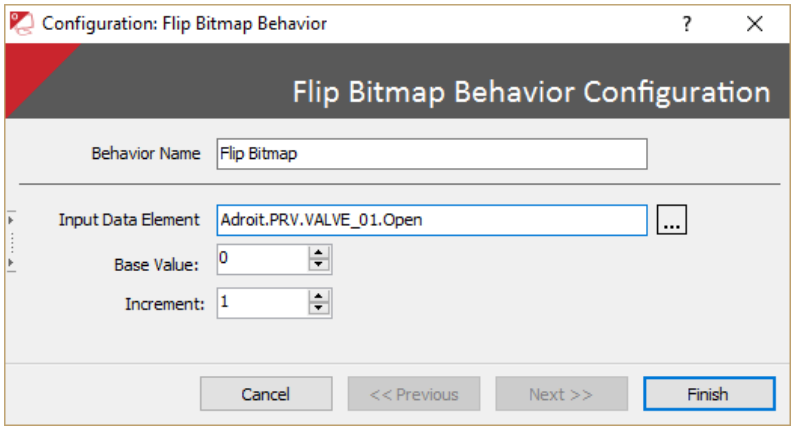
Open/Closed Picture Display

Under the *Vector* group in the Smart UI Designer Toolbox is a *Picture* element which, amongst other things, supports the anachronistically-named *Flip Bitmap* behaviour that shows any one of a number of different pictures based on a tag value.

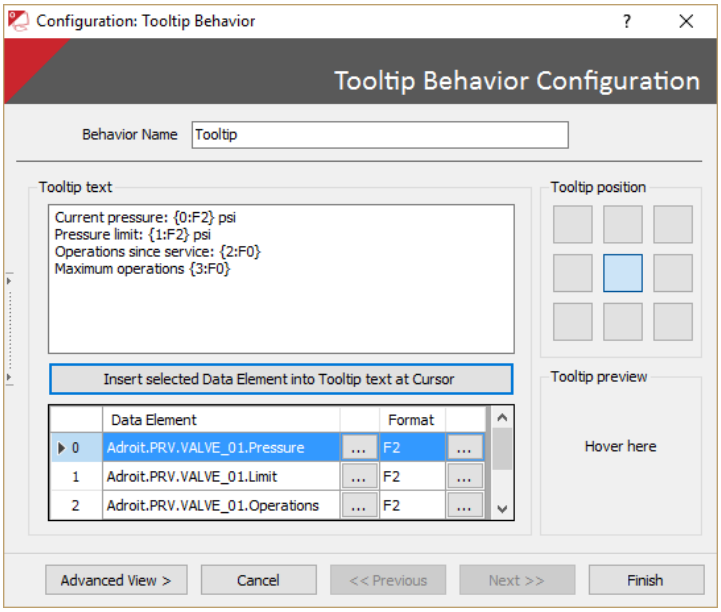


The two images alongside show our PRV in the *Closed* and *Open* positions respectively, and so by editing the *Pictures* property of a *Picture* element we add the two pictures for picture indices 0 and 1 as shown





So copy and paste the two images into the QuickStart Smart UI project folder[1] and add the two pictures to the *Pictures* collection as described above. All that is then necessary to animate the picture in response to changes in valve position is to apply a *Flip Bitmap* behaviour driven by the *Open* slot as shown.



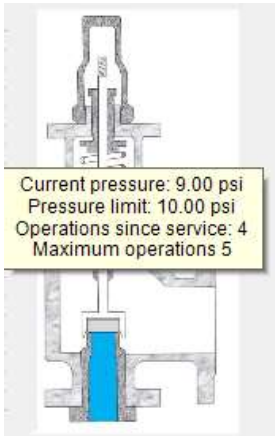
Another useful behaviour that can be applied to our valve picture is *Tooltip.*

The screenshot alongside shows a *Tooltip* behaviour with multi-line text displaying Current pressure, Pressure limit, Operations since service, and Maximum operations. The grid at the bottom defines the *Pressure, Limit, Operations,* and *MaxOperations* (not visible) slots to drive these.

At the top-right we are stipulating that the Tooltip position is middle-centre of the picture.

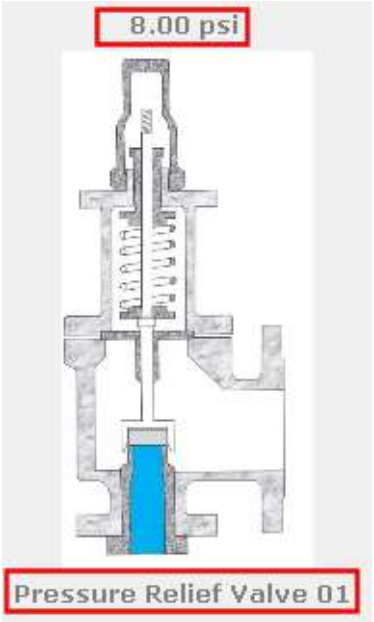
Clicking the *Advanced View* button allows you to specify a hover delay before the tool-tip shows up, and another delay (usually longer) before the tool-tip disappears.

The screenshot below shows hovering over the picture with our valve in the closed position.

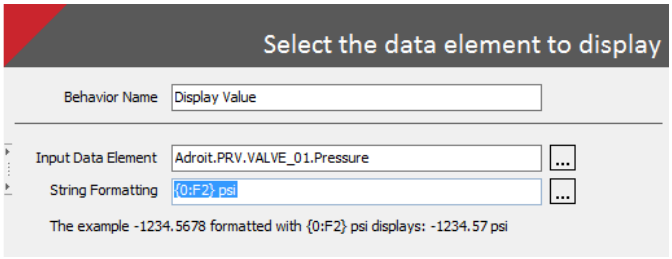


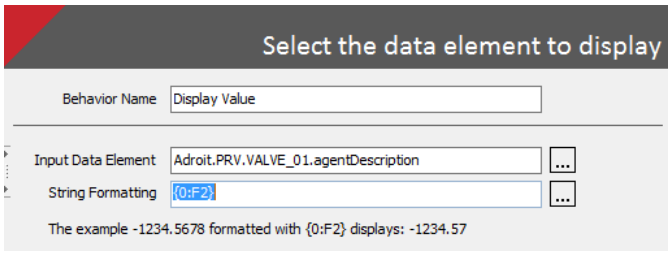
[1] C:\ProgramData\Adroit Technologies\Adroit\SmartUI\Configurations\QuickStart\Projects\Quick Start

Text Displays



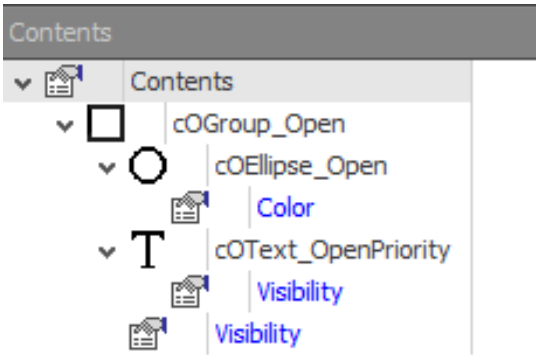
Text elements above and below the picture are used to display the current pressure value and PRV description respectively…





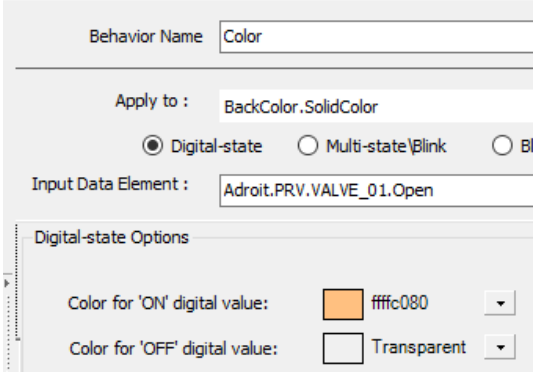
Open Alarm Indication

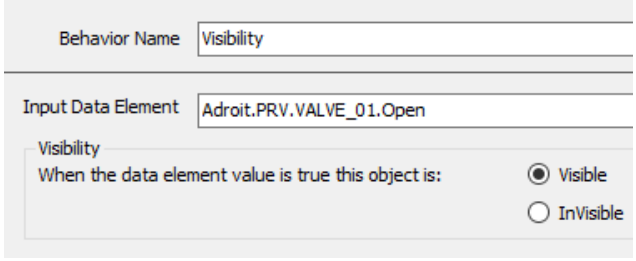
Towards the top-right of our valve picture a number of graphical elements are used to indicate a *Normal* priority *Open* alarm situation.



Firstly we *Group* an Ellipse (circle) to indicate alarm colour and a Text element to display alarm priority. When looking at the group in the *Contents* pane we see that the ellipse has *Colour* behaviour, the Text has a *Visibility* behaviour, and the overall group also has a *Visibility* behaviour.

All behaviours are driven from the *Open* slot of our PRV agent, the colour behaviour being background *Orange* for *Open* and *Transparent* for *Closed.* Both visibility behaviours are configured as visible when the *Open* slot is *True* and invisible when the *Open* slot is *False.*





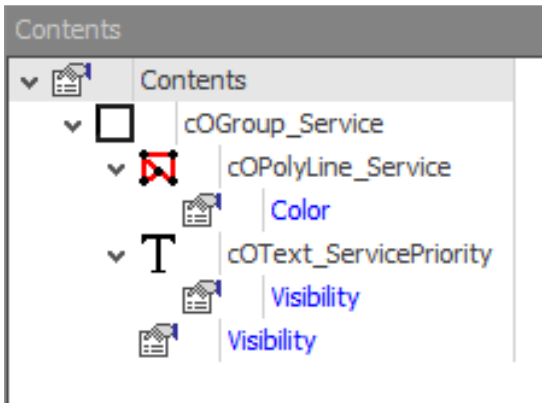
Finally to the right of the ellipse we display a Text element *Open,* visible when the valve is open.

Service Due Alarm Indication

This is considered to be a higher priority alarm than an Open alarm - priority 3 (High) as opposed to priority 2 (Normal)

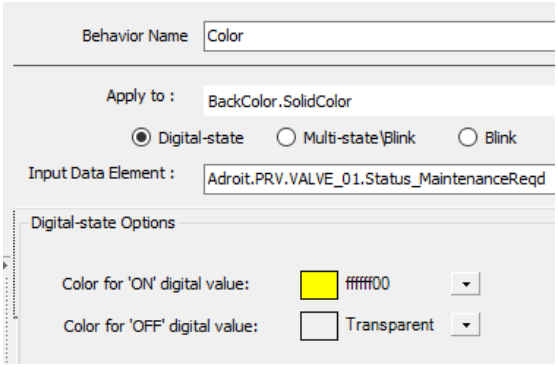
Just below the Open alarm indications we similarly show several graphical elements that indicate a *High* priority *Service Due* alarm situation.

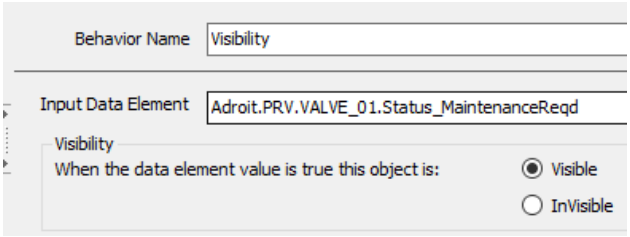
According to ISA 101 situation awareness guidelines this should be indicated by a yellow inverted triangle



Firstly, a group comprising, this time, a closed PolyLine (triangle) to indicate alarm colour and a Text element to display alarm priority. In the *Contents* pane we see that the ellipse has *Colour* behaviour, the Text has a *Visibility* behaviour, and the overall group has a *Visibility* behaviour.

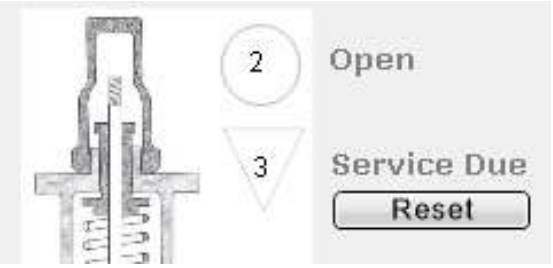
All the behaviours are driven from the *Status_MaintenanceReqd* status bit of our PRV agent. The colour behaviour is background *Yellow* for *ON* and *Transparent* for *OFF.* Both visibility behaviours are configured as visible when the status bit is *ON* and invisible when it is *OFF.*





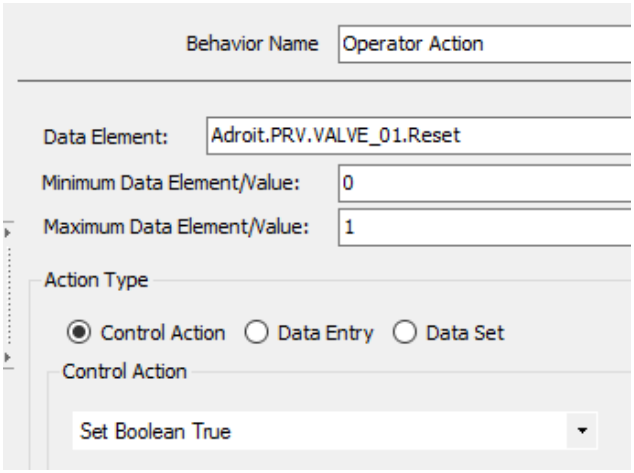
Finally to the right of the ellipse we display a Text element *Service Due,* visible when the status bit is *ON*.

Reset Button



Below the Open and Service Due alarm indications, we locate a Button element captioned *Reset*, the purpose of which is to allow the operator to set a PRV agent's *Reset* slot to ON after maintenance has been successfully completed.

This button need only be visible when maintenance is actually required, so the initial behaviour we configure is a *Visibility* behaviour, driven in the same way as the previous behaviours by the *Status_MaintenanceReqd* status bit of the PRV agent.



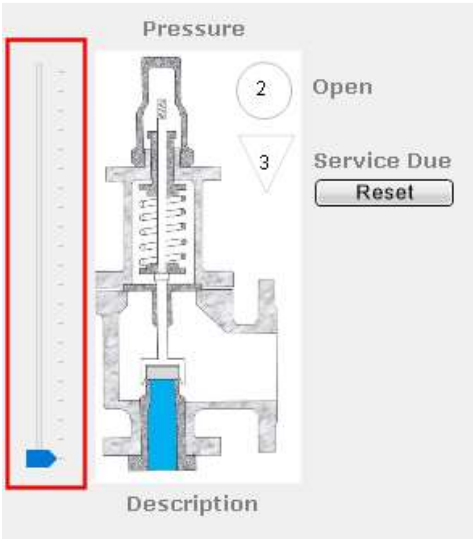
The other behaviour we need to configure for this button is an *Operator Action* behaviour to carry out the actual reset action. This is shown alongside as a *Control Action, Set Boolean True, Operator Action,* operating on the *Reset* slot of the PRV agent

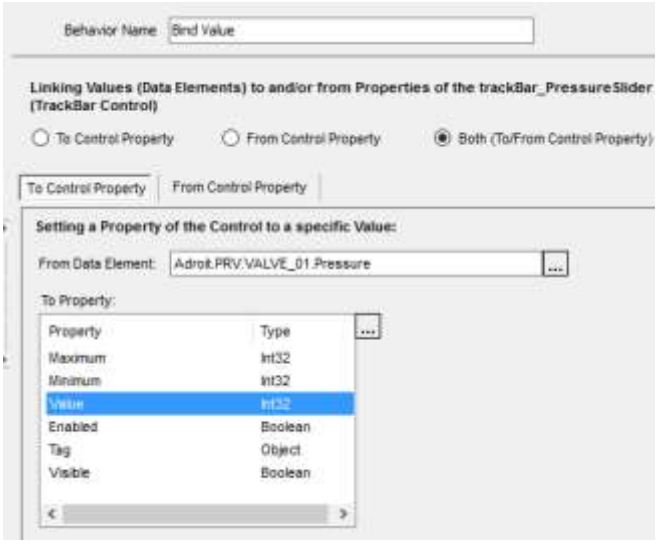
TrackBar Windows Forms Control

In a real-world application, the current pressure value would in all probability be acquired as a process variable by scanning some PLC register. But in our case, to avoid this complexity, and to be able easily to execute various scenarios, we drop a *TrackBar* Windows Forms Control to the left of our PRV picture. Be sure to set its *Orientation* property to *Vertical* since, by default, trackbar controls are horizontally orientated.

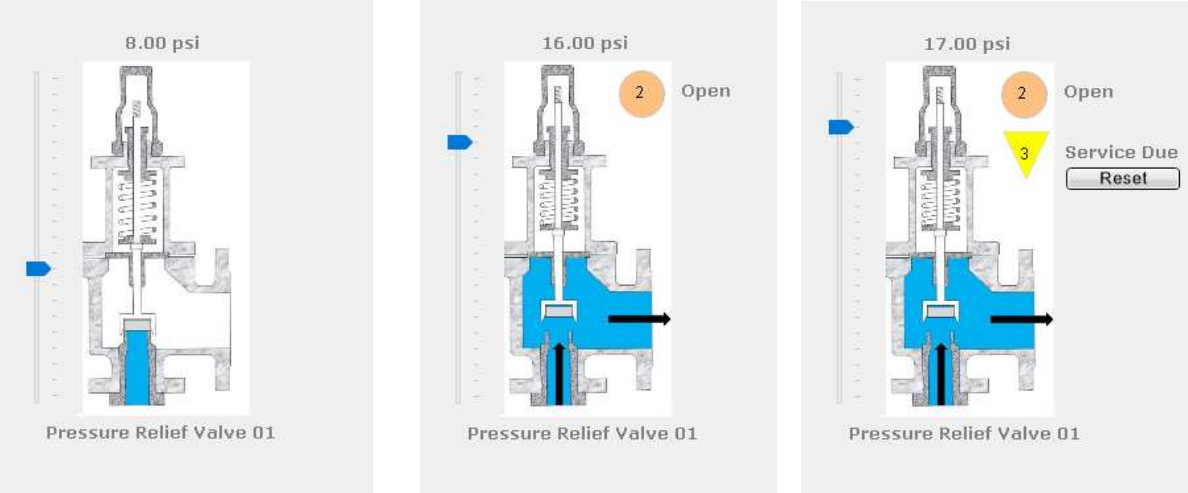
Because we want the TrackBar to both *manipulate* and *indicate* the current pressure value, we apply a *Bind Value* behaviour to this control. As shown, we choose our PRV agent's *Pressure* slot as the *Data Element* to bind in a bi-directional sense with the control's *Value* property.

This bi-directional binding is indicated by the *Both (To/From Control Property)* radio button being selected as indicated near the top-right of the right-hand screenshot below.





Putting it all together



The three screenshots above show our form, with all the graphical elements previously described, configured for a single PRV agent instance VALVE_01.

The leftmost screenshot shows the valve operating normally in the closed position at a pressure of 8.0 psi which is below the 10.0 psi pressure limit.

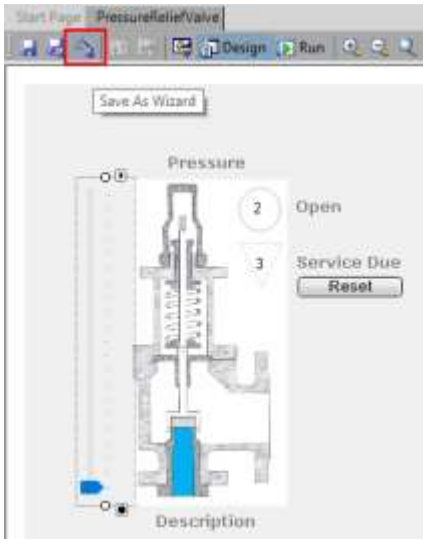
The middle screenshot shows the valve in the open position at 16.0 psi which is above the 10.0 psi pressure limit. A normal priority *Open* alarm is indicated by the orange circle showing alarm priority 2.

The rightmost screenshot shows the valve in the open position at 17.0 psi which is above the 10.0 psi pressure limit. A normal priority *Open* alarm is indicated by the orange circle showing alarm priority 2, but in addition because we have reached the maximum of 5 operations before maintenance is required, a high priority *Service Due* alarm is indicated by the yellow inverted triangle showing alarm priority 3.

Note that in keeping with the ISA 101 situation awareness guidelines, the screenshots are mostly low-impact colours which will not distract an operator. The eye is definitely drawn to the high priority *Service Due* alarm, and to a lesser extent to the normal priority *Open* alarm.

The next step is to transform this hard-coded graphic form into a wizard, but before doing so convince yourself that all the graphic elements and behaviours work as they should do.

Transforming Form into Wizard

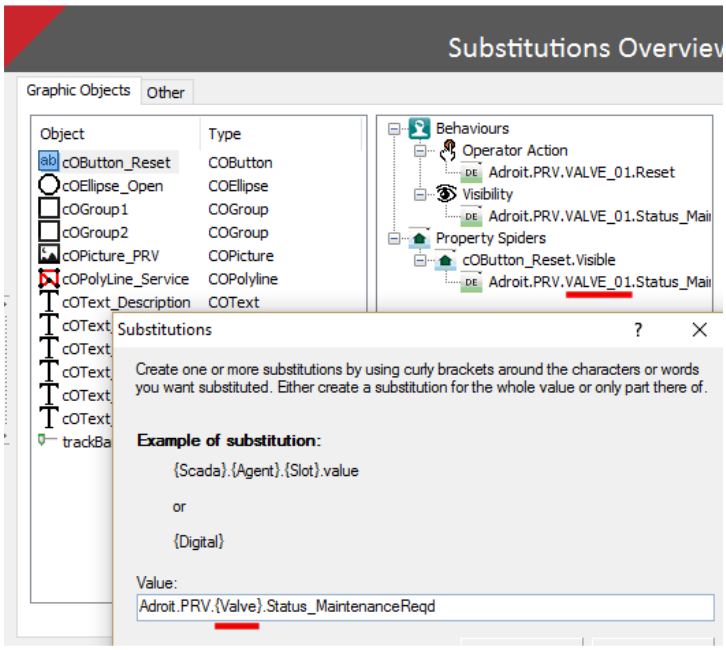


The next step is to transform our graphic form, hard-coded to show only VALVE_01, into a wizard that can potentially be used to visualize any PRV agent.

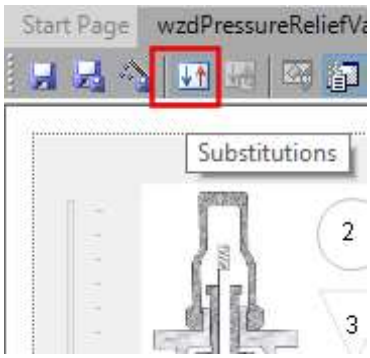
To do this we click the *Save As Wizard* toolbar button as shown. Accept the *Optimise graphic from size* option, and the suggested *wzdPressureReliefValve* wizard form name.

At this point you will presented with a *Substitutions Overview* dialog as shown below. On this dialog you need meticulously to go through every graphic object behaviour, double-clicking and replacing the hard-coded 'VALVE_01' string with a string of the form '{Valve}'.

The curly braces are very important because they are substitution delimiters recognized by Smart UI Designer to replace the placeholder *Valve* string with actual PRV agent names when the wizard is pasted onto a graphic form. It is strongly recommended you copy the '{Valve}' string to the clipboard (Ctrl-C) before carrying out the replacements, and then paste the string back from the clipboard (Ctrl-V) so as to minimize the likelihood of any errors.



If, at any time after saving your form as a wizard, you'd like to go back to the *Substitutions Overview* dialog again, for example to verify that everything has been replaced correctly, you can do so by clicking the *Substitutions* toolbar button as shown.



Once you are satisfied that all VALVE_01 replacements have been successfully carried out, drag *wzdPressureReliefValve* across from *Enterprise Manager* on the left to the *Favourites* pane on the right in readiness for pasting the wizard onto a graphic form.

You may need to enable display of the *Favourites* pane on the right by selecting it from the relevant *View* tab toolbar in Smart UI Designer as shown.

Making use of the wizard

In order to make use of and demonstrate the capabilities of our wizard, in the Configurator window create two more PRV agents named, say, VALVE_02 and VALVE_03. Make sure they have different *Limit* slot and *MaxOperations* slot values as shown below.

Edit PRV Agent : VALVE_01

Slot	Value
Pressure	6.000000
Limit	10.000000
Open	0
Operations	0
MaxOperations	5
Reset	0

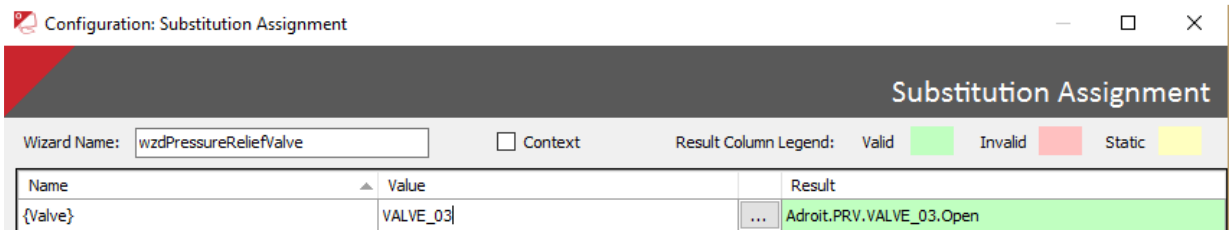
Edit PRV Agent : VALVE_02

Slot	Value
Pressure	13.000000
Limit	15.000000
Open	0
Operations	0
MaxOperations	7
Reset	0

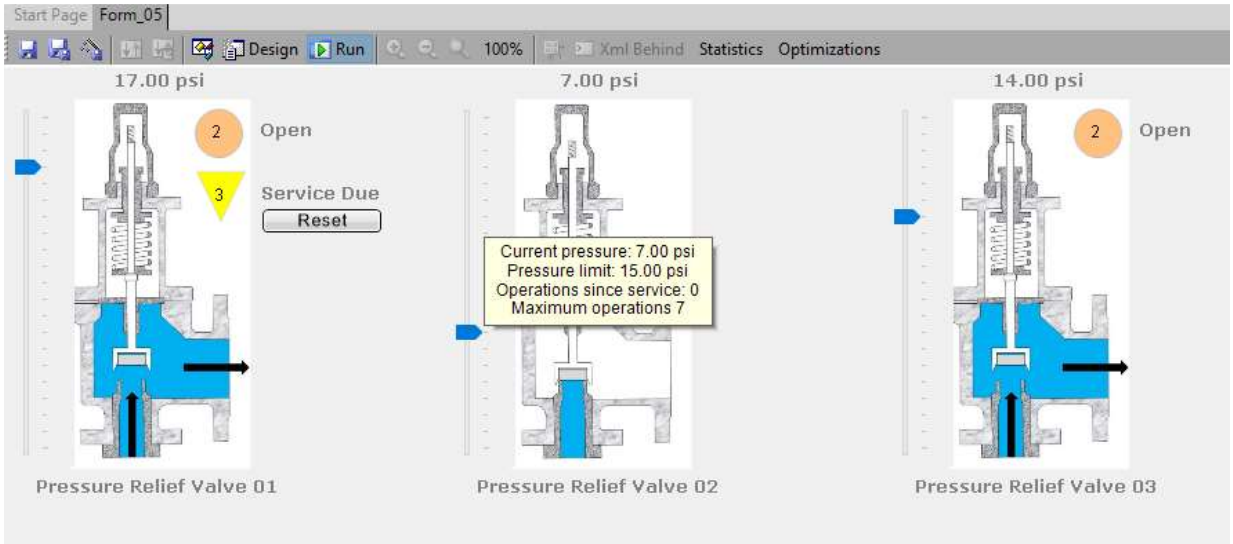
Edit PRV Agent : VALVE_03

Slot	Value
Pressure	7.000000
Limit	7.500000
Open	0
Operations	0
MaxOperations	12
Reset	0

Next create a graphic form Form_05, and drag the wizard across from the *Favourites* pane on the right, drop it onto Form_05 somewhere, and select the *Paste Wizard* context menu item. This will invoke the *Substitution Assignment* dialog shown below, where you can, for example, select VALVE_03 to replace the '{Valve}' placeholder string

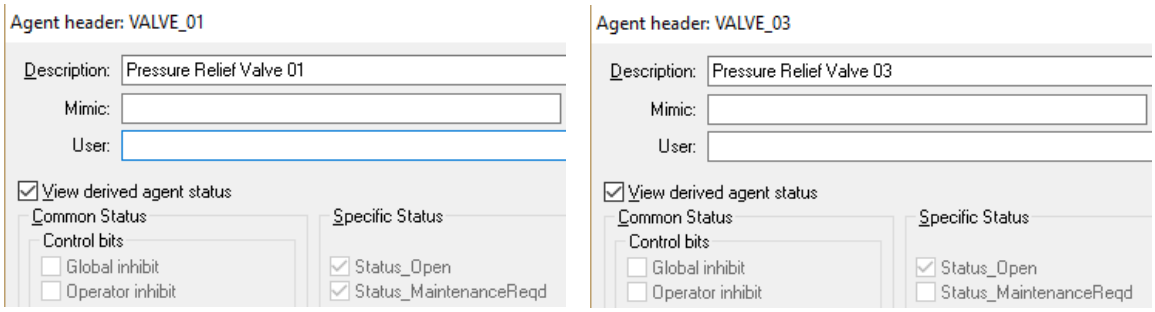


Pasting three instances of our wizard onto Form_05, one for each PRV agent should result in something like the screenshot below. Note that VALVE_01 has an *Open* and *Service Due* alarm, VALVE_02 is operating normally (closed) with its tool-tip displayed, and VALVE_03 has an *Open* alarm but does not yet require maintenance.

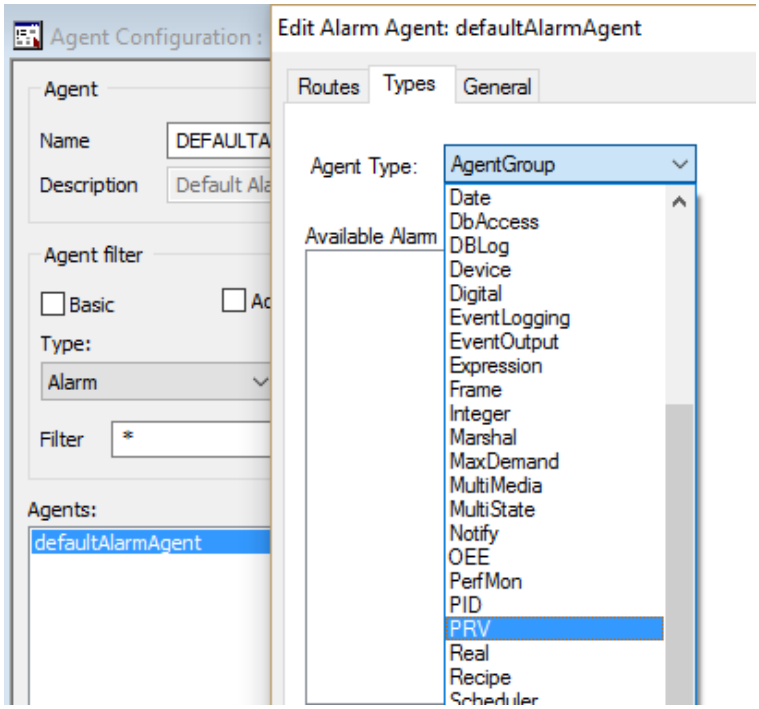


Alarming the PRV Custom Agent

From the graphic form above we can see that VALVE_01 has both an *Open* and *Service Due* alarm and VALVE_03 has just an *Open* alarm. This can also be seen by looking at each agent's header (just click the *Header* button with the required agent selected in the Configurator window) as shown in the screenshots below.



Note the state of the checkboxes in the *Specific Status* section. This is precisely where custom agent development integrates neatly with the Adroit alarming subsystem.



To see this, with all the *Agent filter* checkboxes unchecked in the Configurator window, select *Alarm* agent type, and edit the *defaultAlarmAgent* as shown in the screenshot alongside.

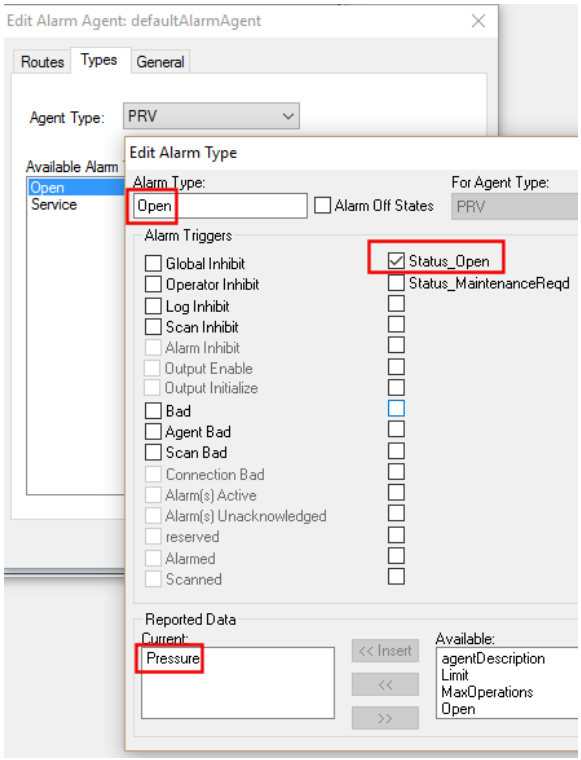
Navigate to the *Types* tab and you will see in the *Agent Type* drop down there is an entry for our newly created PRV agent type.

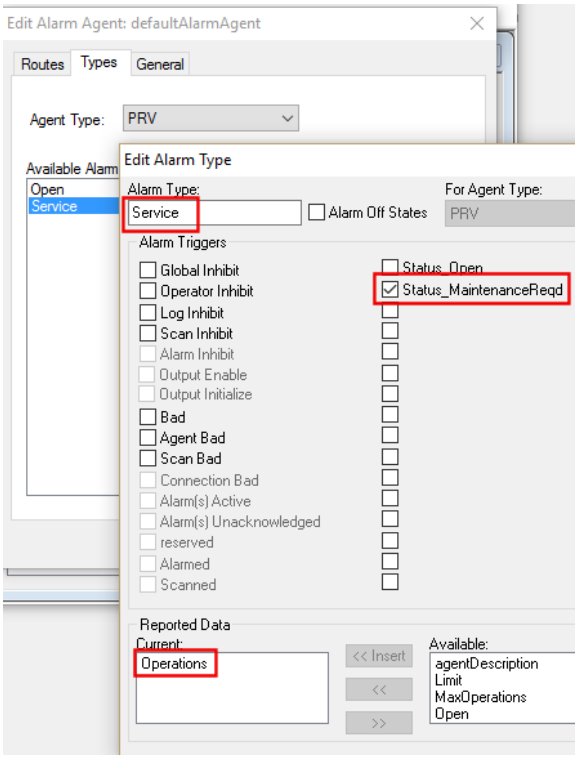
Select PRV agent type, and click the *Add an entry* button above the list of Available Alarm Types which will initially be empty.

The screenshots below show the two alarm types we create for our PRV agent type.

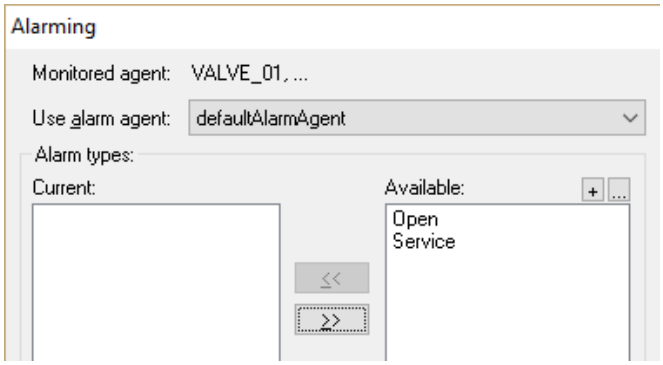
On the left we create an alarm type called *Open* that is raised whenever the *Status_Open* status bit transitions to ON. At the bottom we designate that it's the *Pressure* slot we show in the alarm list's *Reported Data* column for this alarm type, by moving this slot over from the list of *Available* to *Current* slots using the left chevron button.

On the right we create a *Service* alarm type that is raised whenever the *Status_MaintenanceReqd* status bit transitions to ON. In the *Reported Data* section at the bottom we designate the *Operations* slot to be shown in the alarm list's *Reported Data* column for this alarm type.





The final part to alarming our three PRV agents is to multiply select all three of them in the Configurator window, and click the *Alarm* button on the right.



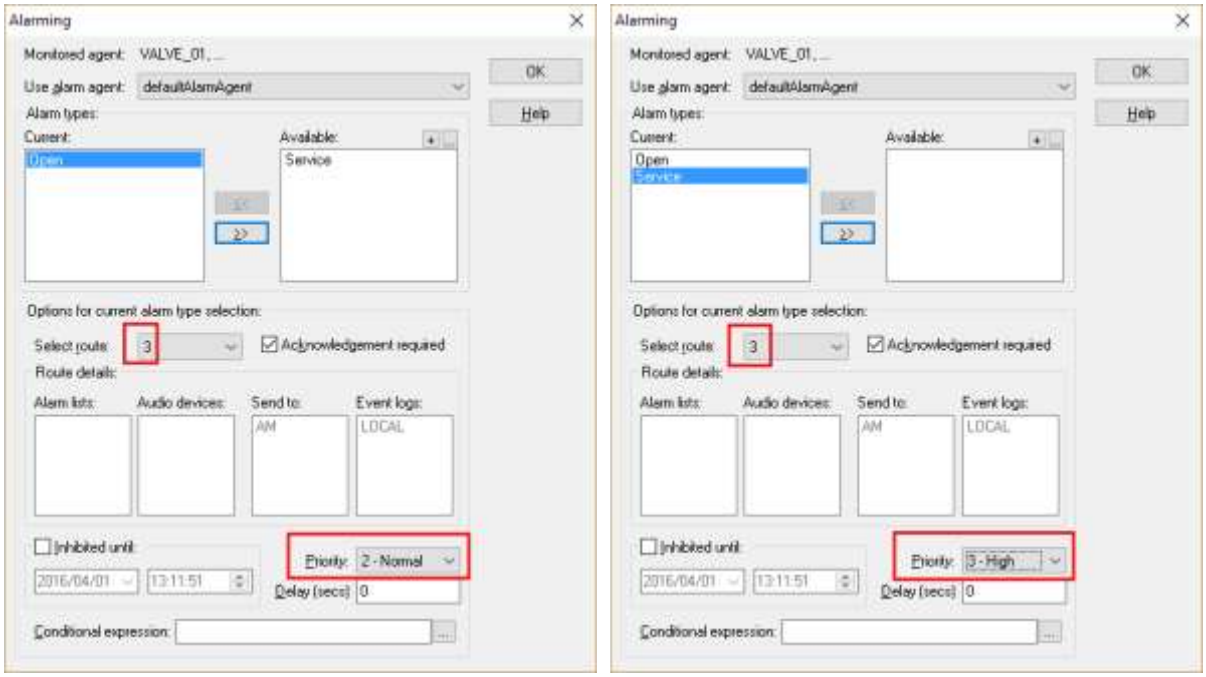
This will display a dialog listing the alarm types available to alarm the PRV agents: *Open* and *Service* as we created previously.

Select the *Open* type and move it across from *Available* to *Current* by clicking the left chevron button.

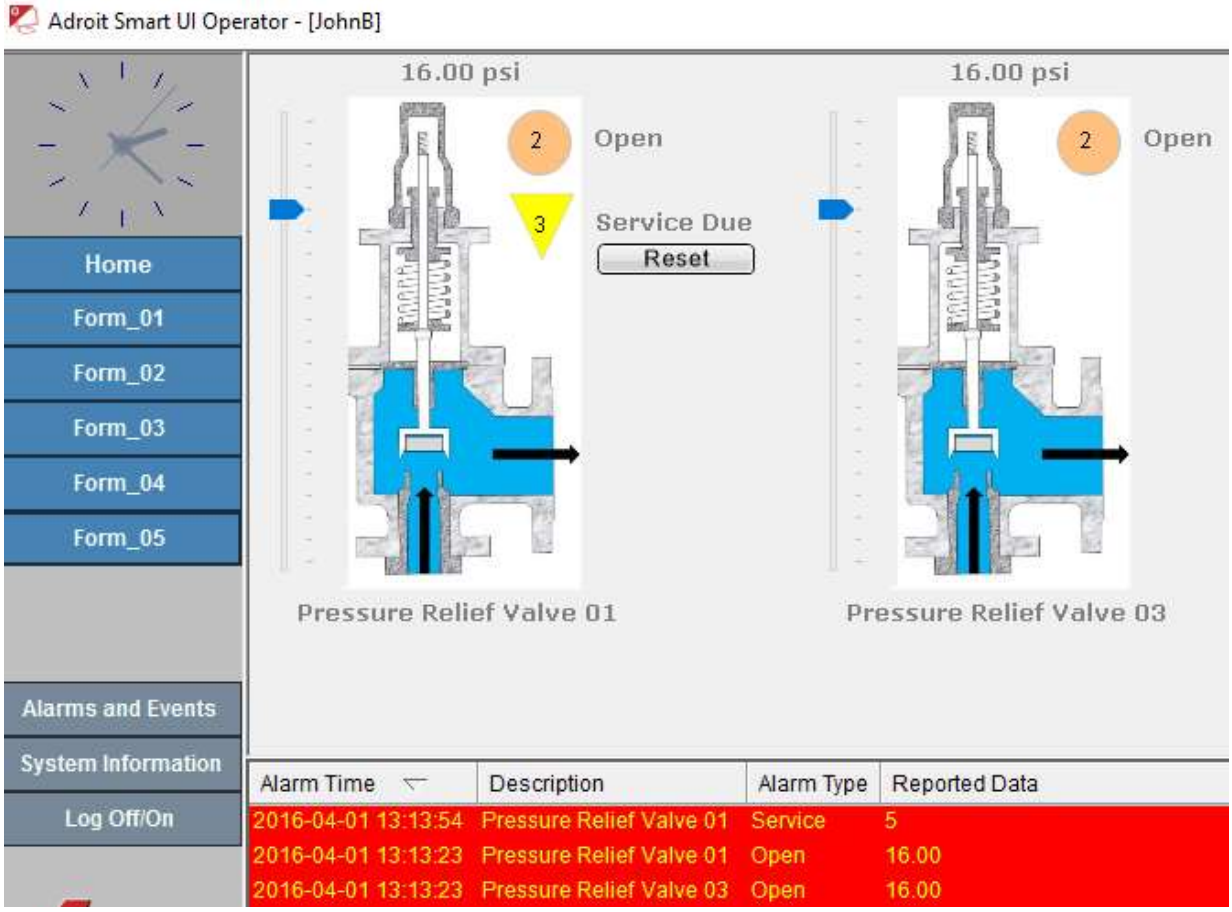
Choose route 3 to avoid the system beeper sounding when an alarm occurs, and choose Priority 2 – Normal for this alarm type.

Next, since we want to see both *Open* and *Service* alarms, select the *Service* alarm type as well, and move it across in the usual way from the list of *Available* to *Current* alarm types. We also use route 3, but we choose Priority 3 – High for this alarm type.

Configuration of both alarm types for all PRV agents is shown in the screenshots below.



In order to visualize our PRV agents and at the same time see alarms in the alarm list, we adapt the *Navigation Template* used in earlier quick start guides by adding a button to display our new Form_05. The result running in Smart UI Operator is shown below.



Conclusion

So, we have created our own custom SCADA database object type and integrated it seamlessly along with all the existing in-built database types. We have proved that it operates in the way that we want it to and gone on to create a comprehensive graphical representation for it. Finally we have accessed the custom alarm indications designed into the new type and smoothly integrated them into the existing Adroit alarm subsystem.

System Integrators and OEMs building their own solutions on top of Adroit, by virtue of this capability, will be able to create whole libraries of data types and corresponding graphical representations, effectively encapsulating their specialized knowledge and know-how of a particular industry sector. These libraries will be usable across countless different project and product configurations, providing significant competitive advantage in terms of both productivity and functionality.

Feel free to email feedback and suggested improvements to support@adroit-europe.com and also to explore the Adroit web-site http://adroit-europe.com and download further training material from ftp://adroit-europe.com/Adroit SmartSCADA Training.pdf