Alarm Areas Example

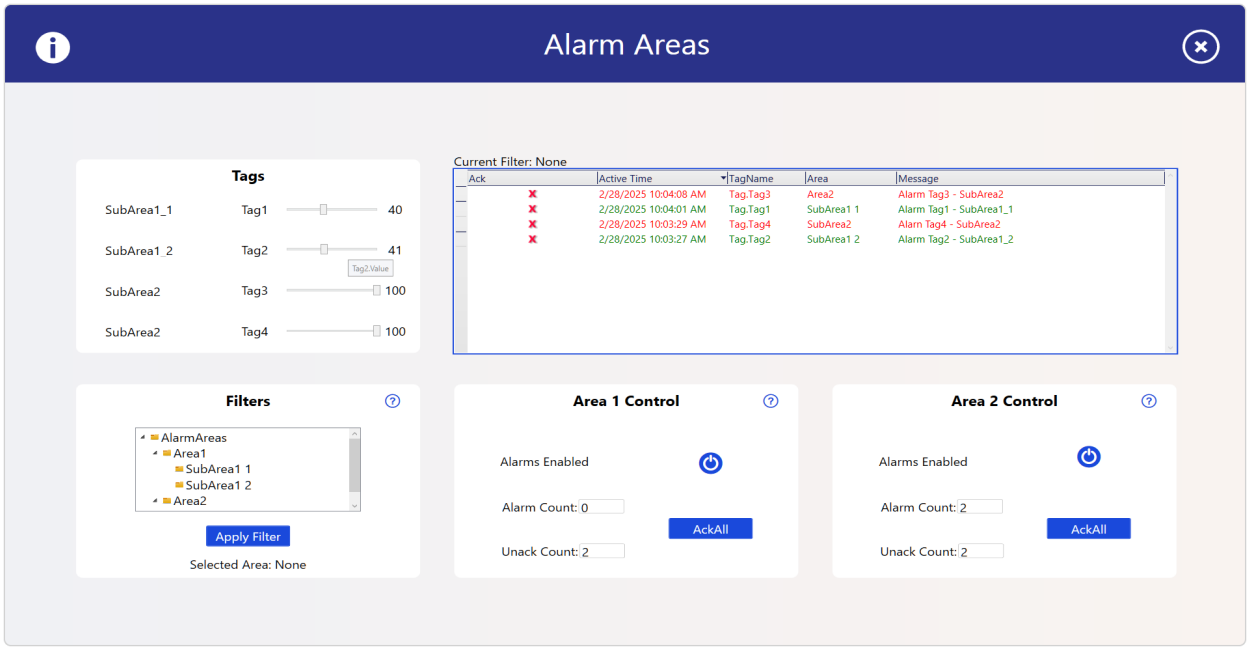
Organize alarms by area.

Download the solution Alarm Areas Example.dbsln.

This example illustrates how to create and use hierarchical `AlarmAreas` in your application.

Summary

This example demonstrates how to configure and visualize hierarchical alarm structures with AlarmAreas. It makes possible to enable, disable, acknowledge, count and filter Alarms based on the Area they are linked to, simplifying the management of complex solutions.



Technical Information

This sample demonstrates how to organize, count, acknowledge, and filter alarms using hierarchical AlarmAreas.

Creating and Applying AlarmAreas

To create an AlarmArea, go to Alarms / Areas. It is possible to create SubAreas by selecting an existing AlarmArea and creating a new one inside of it. In order to apply AlarmAreas, go to Alarms / Items configuration, right-click on the column headers, enable the Area column, and assign each AlarmItem to its corresponding Area by editing the column.

Active Alarms Count by Area

Each AlarmArea maintains its own count of active alarms and provides properties to acknowledge (AckAll, AckAllLocal) or disable all associated alarms. You can link these properties to buttons for alarm management; AckAll and AckAllLocal are change-triggered, so an acknowledge button should use the Toggle action (ActionType ToggleValue) to acknowledge on every press. To explore how it's configured, double-click the buttons in the example using Draw mode and examine their Dynamics and Text properties.

Filter Alarms by Area

Additionally, Areas can be used to filter AlarmItems, simplifying and organizing the contents of AlarmViewers. To do this, go to Displays / Draw / Modules, select an AlarmAreas component and place it on the desired display. Set the SelectedArea field to a text tag, and then apply it to the AlarmAreas through the display's CodeBehind, like this:

```
public async void ApplyFilterButtonDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    if(String.IsNullOrEmpty(@Tag.SelectedArea))
        @Tag.AlarmFilter = "";
    else
        @Tag.AlarmFilter = @Alarm.Area[@Tag.SelectedArea].GetFilterAsync();
}
```

This code retrieves the Area selected by the user through the AlarmAreas component. The GetFilterAsync method then generates a text filter that includes the selected Area and all of its SubAreas. This filter is assigned to the AlarmFilter text tag, which is applied to the AlarmViewer by double-clicking on it in Draw mode and setting its Filter field to {Tag.AlarmFilter}.

In this section...
