



zenon
by COPA-DATA



The background features a series of overlapping, 3D-rendered rectangular blocks in various shades of blue and orange, creating a sense of depth and perspective against a white and dark blue gradient.

Manuel de zenon Élément WPF de zenon

v.8.10



© 2019 Ing. Punzenberger COPA-DATA GmbH

Tous droits réservés.

La distribution et/ou reproduction de ce document ou partie de ce document, sous n'importe quelle forme, n'est autorisée qu'avec la permission écrite de la société COPA-DATA. Les données techniques incluses ne sont fournies qu'à titre d'information et ne présentent aucun caractère légal. Document sujet aux changements, techniques ou autres.

Contenu

1 Bienvenue dans l'aide de COPA-DATA.....	5
2 WPF element	5
3 Basics	6
3.1 WPF in process visualization.....	7
3.2 Referenced assemblies.....	8
3.3 Workflows	10
3.3.1 Workflow with Microsoft Expression Blend.....	11
3.3.2 Workflow with Adobe Illustrator	11
4 Guidelines for designers.....	12
4.1 Workflow with Microsoft Expression Blend.....	12
4.1.1 Create button as an XAML file with Microsoft Expression Blend	12
4.2 Workflow with Adobe Illustrator	16
4.2.1 Bar graph illustration.....	16
4.2.2 WPF export.....	18
4.2.3 Animation in Blend.....	20
5 Directives pour développeurs	24
5.1 Creation of a simple WPF user control with code behind function.....	24
5.2 Debugging the WPF user control in the Runtime	29
5.3 Data exchange between zenon and WPF user controls.....	34
5.3.1 Data exchange using dependency properties	34
5.3.2 Data replacement via VSTA	38
5.4 Access to the zenon (Runtime) object model from a WPF user control	39
5.4.1 Access via VSTA "variable link".....	39
5.4.2 Access via marshaling	44
6 Engineering in zenon.....	47
6.1 CDWPF files (collective files)	48
6.2 create WPF element	49
6.3 Configuration of the linking	49
6.3.1 Properties.....	51
6.3.2 Events	58
6.3.3 Transformation	59
6.4 Validity of XAML Files.....	62

6.5 Pre-built elements.....	63
6.5.1 Analog clock - AnalogClockControl.....	64
6.5.2 Bar graph vertical - VerticalBargraphControl	65
6.5.3 Progress bar - ProgressBarControl	66
6.5.4 COMTRADE-Viewer.....	66
6.5.5 Energy class diagram.....	78
6.5.6 Diagramme de Pareto.....	79
6.5.7 Circular gauge control	82
6.5.8 Diagramme Sankey	86
6.5.9 Temperature indicator - TemperatureIndicatorControl	87
6.5.10 Universal slider - UniversalReglerControl	89
6.5.11 Diagramme en cascade	91
6.5.12 BACnet WPF Control	93
6.6 Display of WPF elements in the zenon web client.....	110
6.6.1 Engineering in the zenon Editor	110
6.6.2 VSTA code (complex)	111
6.6.3 VSTA code (simplified).....	112
6.7 Examples: Integration of WPF in zenon.....	114
6.7.1 Integrate bar graph as WPF XAML in zenon.....	114
6.7.2 Integrate button as WPF XAML in zenon.....	118
6.7.3 Integrate DataGrid Control in zenon	124
6.8 Error handling.....	131

1 Bienvenue dans l'aide de COPA-DATA

TUTORIELS VIDÉO DE ZENON.

Des exemples concrets de configurations de projets dans zenon sont disponibles sur notre chaîne YouTube (https://www.copadata.com/tutorial_menu). Les tutoriels sont regroupés par sujet et proposent un aperçu de l'utilisation des différents modules de zenon. Les tutoriels sont disponibles en anglais.

AIDE GÉNÉRALE

Si vous ne trouvez pas certaines informations dans ce chapitre de l'aide ou si vous souhaitez nous suggérer d'intégrer un complément d'information, veuillez nous contacter par e-mail : documentation@copadata.com.

ASSISTANCE PROJET

Si vous avez besoin d'aide dans le cadre d'un projet, n'hésitez pas à adresser un e-mail à notre service d'assistance : support@copadata.com

LICENCES ET MODULES

Si vous vous rendez compte que vous avez besoin de licences ou de modules supplémentaires, veuillez contacter l'équipe commerciale par e-mail : E-mail sales@copadata.com.

2 WPF element

With the **WPF** dynamic element, valid WPF/XAML files in zenon can be integrated and displayed.

Note: In the zenon Editor, the standard tooltip for the WPF element is not displayed if a **.wpf** file is linked. Furthermore, in zenon Runtime, the zenon tooltip for WPF elements is not supported.



Information

All brand and product names in this documentation are trademarks or registered trademarks of the respective title holder.

3 Basics

XAML

XAML stands for **Extensible Application Markup Language**. The XML-based descriptive text developed by Microsoft defines graphic elements, animations, transformations, displays of color gradients etc. in Silverlight and WPF user interfaces. The use of XAML makes it possible to strictly separate design and programming. The designer prepares, for example, the graphical user interface and creates basic animations that are then used by the developers/project planners who create the application logic.

WPF

WPF stands for **Windows Presentation Foundation** and describes a graphics framework that is part of the Windows .NET framework:

- ▶ WPF provides a comprehensive model for the programmer.
- ▶ XAML describes, based on XML, the interface hierarchy as a markup language. Depending on the construction of the XAML file, there is the possibility to link properties, events and transformations of WPF elements with variables and functions of CD_PRODUCTNAME<.
- ▶ The framework unites the different areas of presentation such as user interface, drawing, graphics, audio, video, documents and typography.

For execution in zenon, Microsoft .NET framework version 3.5 or higher is required.

Information

Transparency

In order for WPF controls in which a transparent background has been defined to also be displayed as transparent, the following must be the case on the computer for both Editor and Runtime:

- ▶ The operating system must be at least Windows 8.1
- ▶ The .NET framework version 4.6 or higher must be installed

WPFs are not shown as transparent in Windows 7 or 8. Instead, the transparent areas are filled with the background color set on the zenon screen.

3.1 WPF in process visualization

XAML makes different design possibilities possible for zenon. Display elements and dynamic elements can be adapted graphically regardless of the project planning. For example, laborious illustrations are first created by designers and then imported into zenon as an XAML file and linked to the desired logic. There are many possibilities for using this, for example:

DYNAMIC ELEMENTS IN ANALOG-LOOK



Graphics no longer need to be drawn in zenon, but can be imported directly as an XAML file. This makes it possible to use complex, elaborately illustrated elements in process visualization. Reflections, shading, 3D effects etc. are supported as graphics. The elements that are adapted to the respective industry environment make intuitive operation possible, along the lines of the operating elements of the machine.

INTRICATE ILLUSTRATIONS FOR INTUITIVE OPERATION



The integration of XAML-based display elements improves the graphics of projects and makes it very easy to display processes clearly. Elements optimized for usability make operation easier. A clear display of data makes it easier to receive complex content. The flexible options for adapting individual elements

makes it easier to use for the operator. It is therefore possible for the project planners to determine display values, scales and units on their own.

CLEAR PRESENTATION OF DATA AND SUMMARIES



Grouped display elements make it possible to clearly display the most important process data, so that the equipment operator is always informed of the current process workflow. Graphical evaluations, display values and sliders can be grouped into an element and make quick and uncomplicated control possible.

INDUSTRY-SPECIFIC DISPLAYS



Elements such as thermometers, scales or bar graphs are part of the basic elements of process visualization. It is possible, using XAML, to adapt these to the respective industry. Thus equipment operators can find the established and usual elements that they already know from the machines in process visualization at the terminal.

ADAPTATION TO CORPORATE DESIGN



Illustrations can be adapted to the respective style requirements of the company, in order to achieve a consistent appearance through to the individual process screen. For example, the standard operation elements from zenon can be used, which can then be adapted to color worlds, house fonts and illustration styles of the corporate design.

3.2 Referenced assemblies

It is not just standard objects (rectangles, graphics, etc.) or effects (color gradients, animations, etc.) that can be displayed using the **WPF elements**, but also customized user controls (with logic in the code behind), which are referenced as assemblies.

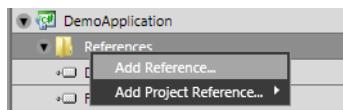
For example, a user control that looks like a tacho and provides special properties and optical effects can be created, such as a **Value** property, which causes the pointer of the tacho to move and/or the corresponding value to be displayed in a label.

Procedure:

- ▶ Draw the appearance of the user controls with standard objects that are offered by WPF.
- ▶ Program the properties and interactions.
- ▶ Compile the complete package.

The result is available as a .NET assembly.

This assembly can be used for WPF projects. To do this, it must be referenced (linked) in the WPF editor (for example: Microsoft Expression Blend). To do this, select the assembly in the zenon file selection dialog:



From this point in time, the WPF user controls of the assembly in the tool box can be selected under **Custom user controls** and used in the WPF project.

Read more about this in the **Guidelines for developers** (à la page 24).

USED REFERENCED ASSEMBLIES IN ZENON

To use an assembly in zenon, this must be provided as a file.

Collective files in **.cdwpf** format manage the assemblies independently. No further configuration is necessary. Assemblies must be added to the *Files* folder for **.xaml** files:

- ▶ Click on *Files* on the project tree.
 - ▶ Select *Others*
 - ▶ Select **Add file...** in the context menu
- The file selection dialog is opened.
- ▶ Insert the desired assembly

When displaying a WPF file in the **WPF element** (Editor and Runtime), the assemblies from this folder are loaded. It is thus also ensured that when the Runtime files are transferred using **Remote Transport**, all referenced assemblies are present on the target computer.

A collective file (**.cdwpf**) can exist alongside an XAML file with the same name. All assemblies (***.dll**) from all collective files and the *Other* folder are copied to the work folder. Only the highest file version is used if there are several assemblies with the same name.

thumb-up Hint

DLLs that are part of a WPF element can also be replaced during ongoing operation. In doing so, the referencing is via linking in the XAML file.

To replace a DLL:

- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In Explorer, replace the DLL in the `\wpfcache` folder of the Editor files.
You can find this folder in the SQL directory under
`...\\PROJECT-GUID\\FILES\\zenon\\custom\\wpfcache`

As an alternative to replacement using Explorer, you can also replace the file in the zenon Editor directly. To do this, carry out the following steps:

- ▶ In the Visual Studio project settings, increase the file version of the DLL.
- ▶ Create the new DLL.
- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In the zenon Editor, delete the DLL from the `\Files\Other` folder and add the file with the higher version number.

MULTI-PROJECT ADMINISTRATION

With multi-project administration, the same assembly must be used in all projects. If an assembly is replaced by another version in a project, it must also be replaced in all other projects that are loaded in the Editor or in Runtime.

3.3 Workflows

The WPF/XAML technology makes new workflows in process visualization possible. The separation of design and functionality ensures a clear distinction of roles between the project engineer and designers; design tasks can be easily fulfilled by using pre-existing designs, which no longer need to be modified by the project engineer.

The following people are involved in the workflow to create WPF elements in zenon:

- ▶ Designer
 - ▶ illustrates elements
 - ▶ takes care of the graphics for MS Expression Design
- ▶ MS Expression Blend operator

- ▶ Animates elements
- ▶ Creates variables for the animation of WPF elements in zenon, which project engineer can access
- ▶ Project engineer
 - ▶ Integrates elements into zenon:
 - ▶ stores logic and functionality

We make a distinction:

- ▶ Workflow with Microsoft Expression Blend (à la page 11)
- ▶ Workflow with Adobe Illustrator (à la page 11)

3.3.1 Workflow with Microsoft Expression Blend

When using Microsoft Expression Blend, a WPF element is created in four stages:

1. Illustration of elements in **MS Expression Blend** (à la page 12)
2. Open element in **MS Expression Design** and export as WPF
3. Animation in **MS Expression Blend** (à la page 12)
4. Integration into zenon (à la page 118)

You can find an example for creating a WPF elements with Microsoft Expression Blend in the Create button as XAML file with Microsoft Expression Blend (à la page 12) chapter.

3.3.2 Workflow with Adobe Illustrator

Based on traditional design processes with **Adobe Illustrator** the following workflow is available:

1. Illustration of elements in **Adobe Illustrator** (à la page 16)
2. Import of .ai files and preparation in **MS Expression Design** (à la page 18)
3. WPF export from **MS Expression Design** (à la page 18)
4. Animation in **MS Expression Blend** (à la page 20)
5. Integration into zenon (à la page 114)

You can find an example for creation in the Workflow with Adobe Illustrator (à la page 16) chapter.

4 Guidelines for designers

This section informs you how to correctly create WPF files in Microsoft Expression Blend and Adobe Illustrator. The tutorials on Creating a button element (à la page 12) and a bar graph element (à la page 16) show you how fully functional WPF files for zenon can be created from pre-existing graphics in a few steps.

The following tools were used for this:

- ▶ Adobe Illustrator CS3 (AI)
- ▶ Microsoft Expression Design 4 (ED)
- ▶ Microsoft Expression Blend 4 (EB)
- ▶ zenon



Information

If referenced objects (assemblies) are used in WPF, note the instructions in the Referenced objects (à la page 8) chapter.

4.1 Workflow with Microsoft Expression Blend

With Microsoft Expression Blend, a WPF element:

- ▶ is illustrated
- ▶ is converted into WPF format using **MS Expression Design**
- ▶ animated

The following example shows the illustration and conversion of a button element into an XAML file.

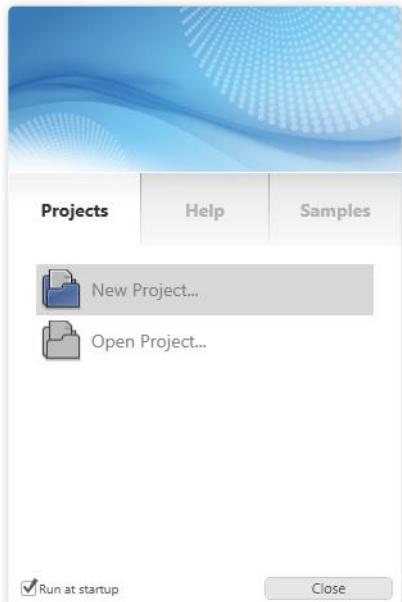
Note: A test version of "Microsoft Expression Blend" can be downloaded from the Microsoft website.

4.1.1 Create button as an XAML file with Microsoft Expression Blend

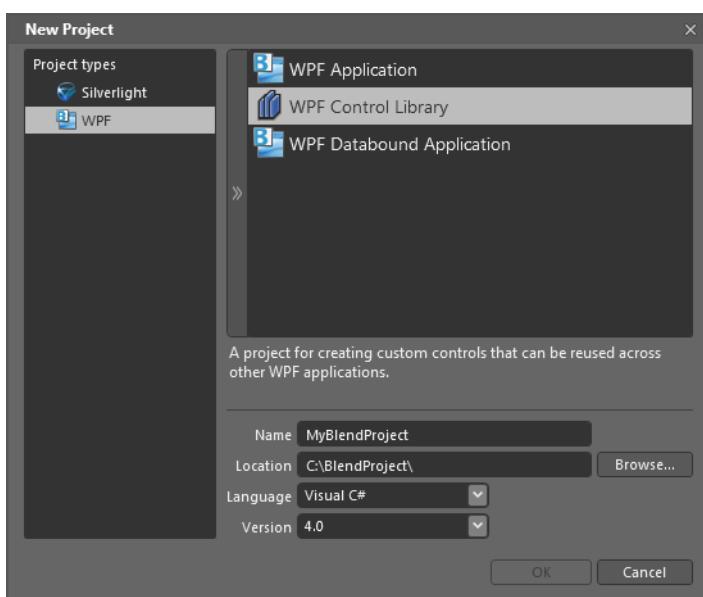
CREATE BUTTON

1. Start Expression Blend

2. select the **New Project** option



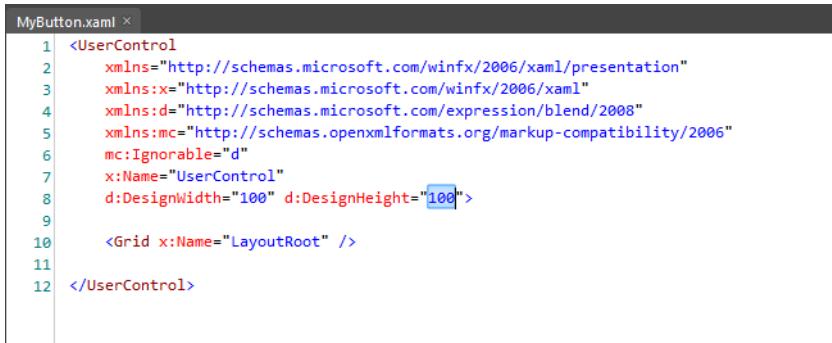
3. Select *WPF* as project type
4. give it a path and name of your choice (MyBlendProject, for example)



The **Language** and **Version** settings can be ignored, because no functionality is to be programmed.

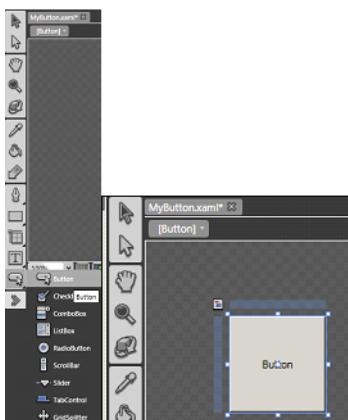
5. After the dialog has been confirmed with **OK**, Microsoft Blend creates a new project with the chosen settings. Expression Blend adds an empty XAML file which already contains a class reference.
6. Delete the CS file that belongs to the XAML file using the context menu.

7. Rename the XAML file **MainControl.xaml** to **MyButton.xaml**.
8. The development size of the file is set at 640 x 480 pixels as standard and must still be changed:
 - a) switch to **XAML** view
 - b) correct the size to 100 x 100 pixels
 - c) Delete the class reference `x:Class="MyBlendProject.MyButton"`



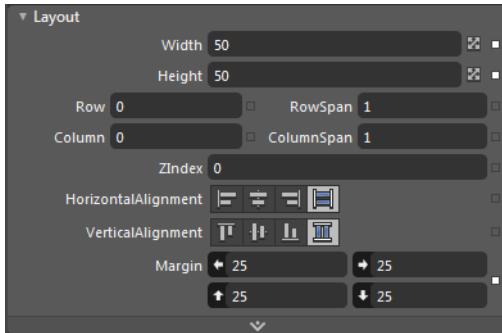
```
MyButton.xaml ×
1 <UserControl
2   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6   mc:Ignorable="d"
7   x:Name="UserControl"
8   d:DesignWidth="100" d:DesignHeight="100">
9
10  <Grid x:Name="LayoutRoot" />
11
12 </UserControl>
```

9. switch to **Design** view

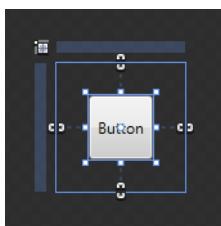


10. add a button via the toolbar
11. define the properties
 - ▶ Width: 50
 - ▶ Height: 50

- ▶ Margins: 25



The button is therefore at the center of the control.



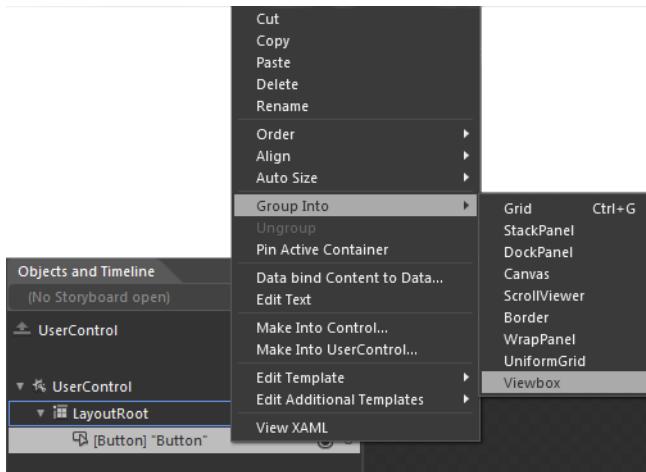
12. Save the changes and open the file in Internet Explorer to check it. You will see that the button is displayed in a size of 50 x 50 pixels.

MAKE BUTTON SCALABLE

If you integrate this status into zenon, the button will always have the exact size of 50 x 50 pixels. Because the button can be implemented as a scalable button, switch to Expression Blend again:

1. Select the button in the tree view.
2. select the **Group Into->Viewbox** button in the context menu
3. the button is inserted into a **Viewbox**
4. Define the properties of the viewbox
 - ▶ Width: Auto
 - ▶ Height: Auto

5. save the file



6. If you now open the file in Internet Explorer, the button is automatically scaled when the IE window size is changed. This file will now also automatically adapt to changes in the size of the **WPF element** in zenon.

CHANGE NAME

Before you can integrate the file into zenon, you must give the **WPF element** a name. The **WPF elements** are not named in Expression Blend as standard, and are labeled with square brackets and their type. zenon content is assigned to WPF content via the name of the **WPF elements**:

- ▶ in tree view, change the name
 - ▶ of the button on **MyButton**
 - ▶ of the ViewBox to **MyViewBox**

This button can now be integrated in zenon (à la page 118) as an XAML file.

4.2 Workflow with Adobe Illustrator

When **Adobe Illustrator** is used, a WPF element:

- ▶ is illustrated in **Adobe Illustrator**
- ▶ is converted into a WPF in **MS Expression Design**
- ▶ is animated in **MS Expression Blend**

The following example shows the illustration and conversion of a bar graph element into an XAML file.

4.2.1 Bar graph illustration

A bar graph is created in Adobe Illustrator.

1. AI: Starting element for bar graph



Illustrated in Adobe Illustrator CS3.

2. AI: Path view of bar graph in Adobe Illustrator



- ▶ All effects must be converted (**Object -> Convert appearance**)
- ▶ All lines are transformed into paths (**Object -> Path -> Contour line**)
- ▶ Do not use filters such as shading, blurring etc.

NOTES ON COMPATIBILITY

Illustrations that were created with Adobe Illustrator are in principle suitable for WPF export. However, not all Illustrator effects can become corresponding effects in Expression Design/Blend. Note:

Effect	Description
Clipping masks	<p>Clipping masks created in Adobe Illustrator are not correctly interpreted by Expression Design. These are usually shown in Blend as areas of black color.</p> <p>We recommend creating illustrations without clipping masks.</p>
Filters and effects	<p>Not all Adobe Illustrator filters are transferred into Expression Design accordingly: Thus blurring filters, shading filters and corner effects from Illustrator do not work in Expression Design.</p> <p>Solution:</p> <ul style="list-style-type: none"> ▶ Most effects can be converted so that they can be read correctly by Expression Design using the Object -> Convert appearance command in Adobe Illustrator. ▶ Corner effects from Adobe Illustrator are correctly interpreted by MS Design if they are converted to AI in paths.
Text fields	<p>To be able to link text fields with code, these must be created separately in Expression Blend. "Labels" are required for dynamic texts; simple "text fields" are sufficient for static information.</p>

Effect	Description
	There is no possibility to create text labels in MS Design. These must be directly created in MS Blend.
Transparencies and group transparencies	<p>There can be difficulties in Adobe Illustrator with the correct interpretation of transparency settings, in particular from group transparency settings.</p> <p>However MS Expression Blend and MS Expression Design do offer the possibility to create new transparency settings.</p>
Multiply levels	<p>These level settings in Adobe Illustrator are not always correctly displayed by MS Expression Blend.</p> <p>However, there is the possibility to "Multiply levels" directly in Expression Design.</p>
Indicating instruments and standard positions	<p>To prepare the graphics optimally for animation, the indicator and slider must always be set to the starting position, usually 0 or 12:00 o'clock.</p> <p>Thus the position parameters for rotations etc. are also correct in Blend and an animation can be implemented without conversion of position data.</p>

4.2.2 WPF export

WPF files are required for animation in Microsoft Expression Blend. We recommend Microsoft Expression Design for this export, because it provides good results and most Illustrator effects are correctly interpreted.

Note: There is a free plug-in for the direct export of WPF files from Adobe Illustrator available on the internet. This plug-in provides a quick, uncomplicated way of exporting from Illustrator, however it is less suited to the current application because it lead to graphical losses. Even color deviations from the original document are possible.

Files in **.ai** format can regularly be imported into Expression Design; the paths are retained in the process.

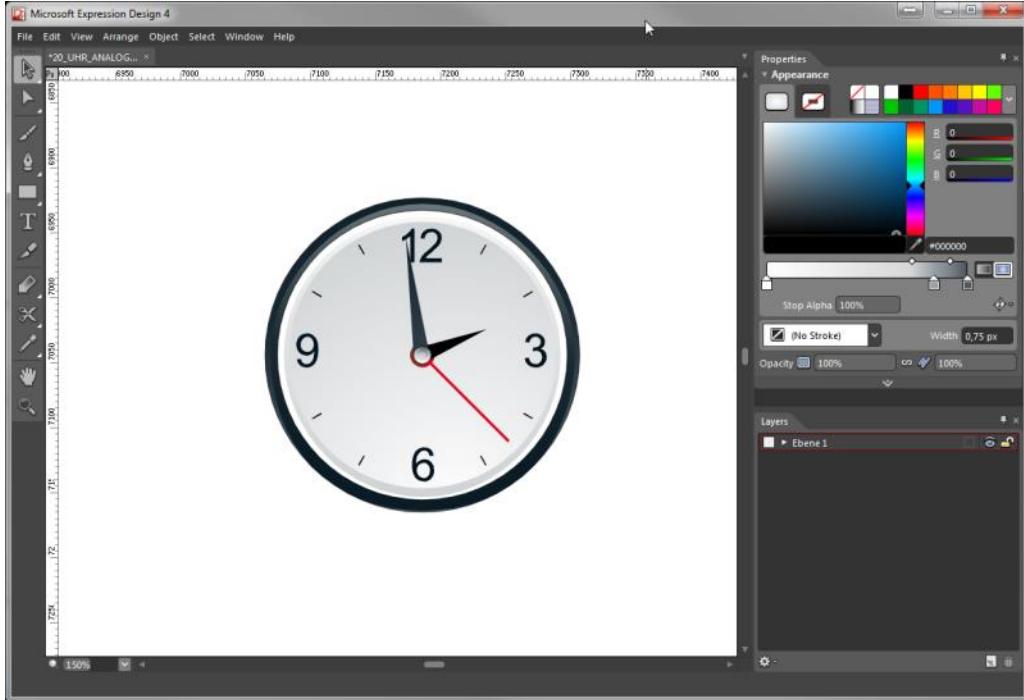
Attention: Some common Illustrator effects cannot be displayed by Expression Design correctly however (see Illustration (à la page 16) chapter).

We export the pre-created bar graph element in 5 stages:

1. **ED: Import**

- ▶ Import the prepared Illustrator file (à la page 16) in **Microsoft Expression Design** via **File -> Import**

2. **ED: Optimization**



- ▶ If the starting file is not correctly displayed in MS Expression Design, it can still be subsequently edited and optimized here

3. **ED: Select**



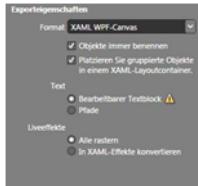
- ▶ Highlight the element for WPF export with the **direct selection** arrow in MS Expression Design; in this case it is the whole clock

4. **ED: Start export**



- ▶ Start the export via **File -> Export**
- ▶ the dialog for configuring the export settings opens

5. **ED: Export settings**



- ▶ Enter the following export settings:
 - a) **Format:** *XAML Silverlight 4 / WPF Canvas*
 - Always name objects:** Activate with tick
 - Place the grouped object in an XAML layout container:** Activate with tick
 - b) **Text:** *Editable text block*
 - c) **Line effects:** *Rasterize all*

The exported file has **.xaml** file suffix. It is prepared and animated (à la page 20) in MS Expression Blend in the next stage.

4.2.3 Animation in Blend

With MS Expression Blend:

- ▶ static XAML files from MS Expression Design are animated
- ▶ Variables for controlling effects that can be addressed by zenon are created

In thirteen steps, we go from a static XAML to an animated element, that can be embedded in zenon:

1. **EB:create project**



- a) Open Microsoft Expression Blend
- a) Create a new project
- b) Select the **Project type** of **WPF- > WPF Control Library**
- c) Give it a name (in our tutorial: **My_Project**)
- d) Select a location where it is to be saved
- e) Select a language (in our tutorial: C#)
- f) Select Framework Version 3.5

2. **EB: delete MainControl.xaml.cs**



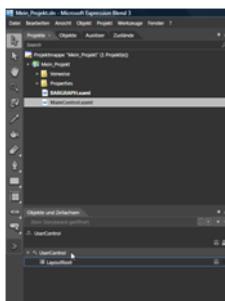
- Navigate to **MainControl.xaml.cs**
- Delete this file using the **Delete** command in the context menu

3. **EB: Open exported XAML file**



- Open the context menu for **My_Project** (right mouse button)
- Select **Add existing element...**
- Select the XAML file exported from Microsoft Expression Design, in order to open this in Microsoft Expression Blend

4. **EB: Open MainControl.xaml**



- Open the automatically created **MainControl.xaml**
- In the **Objects and Time axes** area, navigate to the **UserControl** entry

5. **EB: Adapt XAML code**



- Click on **UserControl** with the right mouse button
 - Select **Display XAML** in the contextual menu.
 - Delete lines 7 and 9 in the XAML code:
- ```
x:Class="My_Project.MainControl"
d:DesignWidth="640" d:DesignHeight="480"
```

## 6. EB: check XAML code



- ▶ The XAML code should now look like this:

```
<UserControl
```

```
 xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
 xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
 xmlns:d=http://schemas.microsoft.com/expression/blend/2008
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
 mc:Ignorable="d"
 x:Name="UserControl">
```

```
 <Grid x:Name="LayoutRoot"/>
 </UserControl>
```

## 7. EB: Copy elements



- Open the XAML file imported from Expression Design
- Mark all elements
- Select **Delete** in the context menu
- Change back to the automatically created XAML file

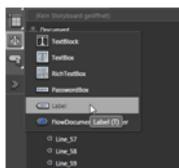
## 8. EB: Insert element



- Click on **Layout Root** with the right mouse button
- Select **Insert**

9. **EB: Adapt layout type**

- a) Click on **Layout root** -> **Change layout type** -> **Viewbox** with the right mouse button
- b) The structure should now look like this: **UserControl** -> **LayoutRoot** -> **Grid** -> **Elements**
- c) Assign a name for **LayoutRoot** and **Grid** by double-clicking on the names

10. **EB: Texts and values**

- ▶ Dynamic and static texts are labeled with text fields
- ▶ Values (numbers) are issued with **Labels**

11. **EB: Insert labels**

- ▶ Labels replace numbers that are to be subsequently linked using INT variables (must be carried out for all number elements)

12. **EB: Set property**

- ▶ To display 100%, set the bar graph element's **MaxHeight** property to 341 (the maximum height of the indicator element is 340)

13. **EB: prepare for use in zenon**

- a) Delete all name labels (names may only be given for elements that are to be addressed via zenon)
- b) Save the XAML file with any desired name

- c) Integrate the XAML file into zenon (à la page 114)

**A tip for checking:** If the XAML file is displayed with no problems in Microsoft Internet Explorer and the window size of Internet Explorer adapts to it, it will also be correctly used in zenon.

## 5 Directives pour développeurs

Cette section décrit la création de contrôles utilisateur WPF simples dotés de fonctionnalités code-behind dans Microsoft Visual Studio, ainsi que le débogage de ces contrôles utilisateur dans le Runtime.

Les outils suivants ont été utilisés à cette fin :

- ▶ Microsoft Visual Studio 2015
- ▶ zenon

### Information

Recommandation : Utilisez Microsoft Visual Studio version 2012 ou une version ultérieure. Le concepteur XAML est mieux intégré dans ces versions.

### 5.1 Creation of a simple WPF user control with code behind function

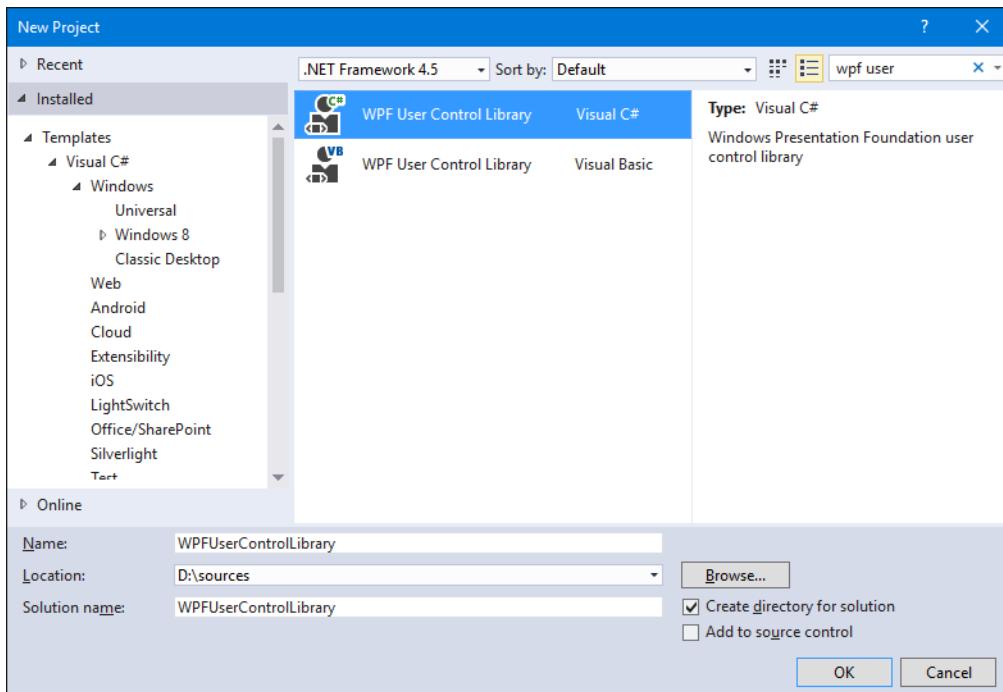
The creation and incorporation of a simple user control is described in this chapter. Because only the fundamental mechanisms/process for integration into zenon is described, the functionality of the user control is limited to the addition of two values. There is intentionally no enhanced error handling or explicit completion, in order to retain the simplicity of this example.

#### CREATE WPF USER CONTROL

1. Create a new **Solution** and a **WPF User Control Library** in this in Visual Studio.

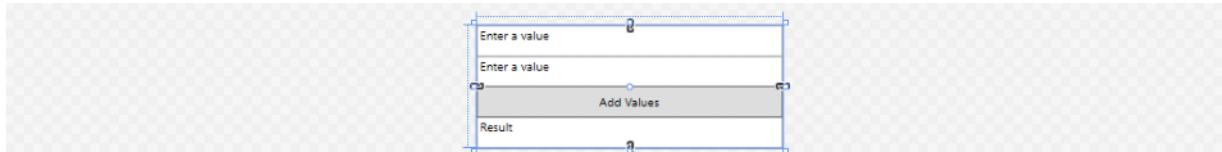
The .NET framework version 4 was selected for this example. A different version can also be selected, which must be installed on the target system on which Runtime will subsequently be started.

**Info:** If the corresponding project template does not appear in the list of available templates, this can be added by means of the search (field at the top right of the dialog).



In our example, the project is given the name **WPFUserControlLibrary**.

2. Create 3 text boxes and a button in the `UserControl1.xaml` file:



```

<UserControl x:Class="WPFUserControlLibrary.UserControl1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:WPFUserControlLibrary"
 mc:Ignorable="d"
 d:DesignHeight="120" d:DesignWidth="300">
 <Grid Height="120" Width="Auto">
 <StackPanel>
 <TextBox x:Name="textBoxA" Height="30" TextWrapping="Wrap" Text="Enter a value" HorizontalAlignment="Stretch"/>
 <TextBox x:Name="textBoxB" Height="30" TextWrapping="Wrap" Text="Enter a value" HorizontalAlignment="Stretch"/>
 <Button x:Name="buttonAdd" Height="30" Content="Add Values" HorizontalAlignment="Stretch" VerticalAlignment="Top" Click="buttonAdd_Click"/>
 <TextBox x:Name="textBoxC" Height="30" TextWrapping="Wrap" Text="Result" HorizontalAlignment="Stretch" IsReadOnly="True"/>
 </StackPanel>
 </Grid>
</UserControl>

```

3. Add the following code in the click event of the button:

```

private void buttonAdd_Click(object sender, RoutedEventArgs e)
{
 try
 {
 textBoxC.Text = (Convert.ToInt32(textBoxA.Text) + Convert.ToInt32(textBoxB.Text)).ToString();
 }
 catch (Exception ex)
 {
 textBoxC.Text = "Error adding values: " + ex.Message;
 }
}

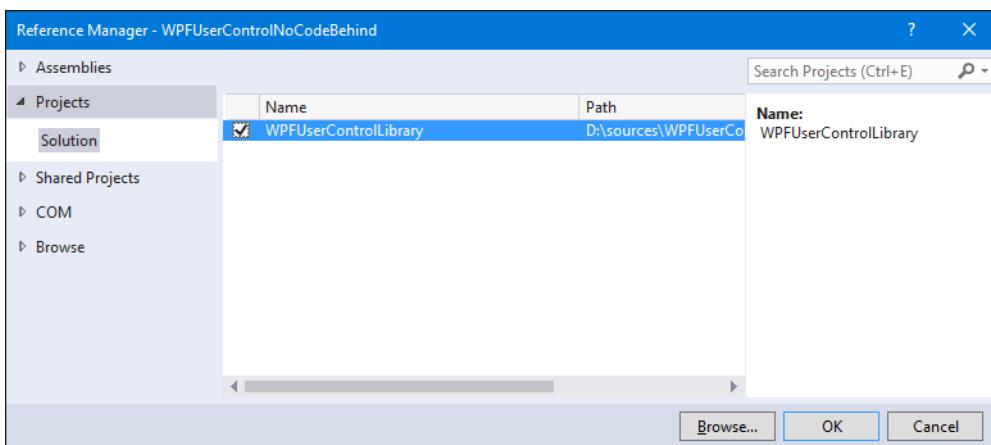
```

Now you have the user control with the required functionality available. However, because zenon can only display XAML files that do not link to a code-behind file, an additional XAML file is needed that references the library (assembly) that has just been built.

## CREATION OF THE XAML FILE (WITHOUT CODE BEHIND) FOR ZENON

Proceed as follows to create the XAML file required in zenon.

1. Create a further project, again as a **WPF User Control Library**
2. It was called **WPFUserControlNoCodeBehind** in our example.
3. Insert a reference to the project that has just been built into this new project.



4. The XAML files (**UserControl1.xaml**) looks as follows:

```
<UserControl x:Class="WPFUserControlNoCodeBehind.UserControl1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:WPFUserControlNoCodeBehind"
 mc:Ignorable="d"
 d:DesignHeight="300" d:DesignWidth="300">
 <Grid>
 </Grid>
</UserControl>
```

5. Because all necessary content is contained in the DLL that has been created and no code-behind file can be used, delete the following lines:

`x:Class="WPFUserControlNoCodeBehind.UserControl1"`  
`xmlns:local="clr-namespace:WPFUserControlNoCodeBehind"`

6. Also delete (for the designer's size setting) the following lines:
- `mc:Ignorable="d"`  
`d:DesignHeight="300" d:DesignWidth="300"`

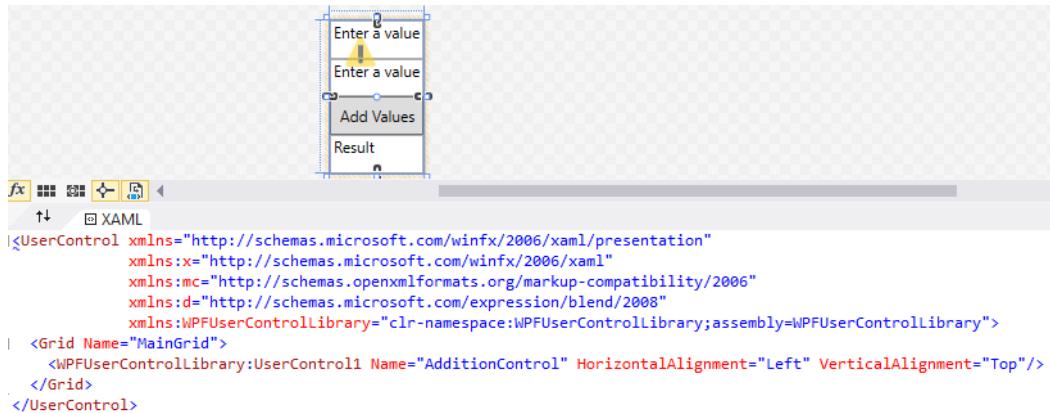
7. Delete the code-behind file (**UserControl1.xaml.cs**) in this project.

8. Drag the user control that has been created beforehand (for the project **WPFUserControlLibrary**) over the toolbox in the XAML designer.

9. Assign a name for the grid and the user control.

**Attention:** If no name is given here, these elements do not appear in the linking dialog in the zenon Editor and thus cannot be made dynamic.

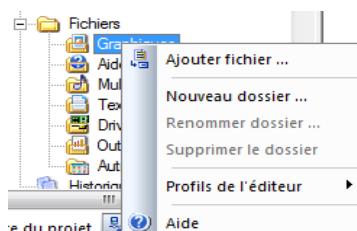
10. The XAML file should now look as follows:



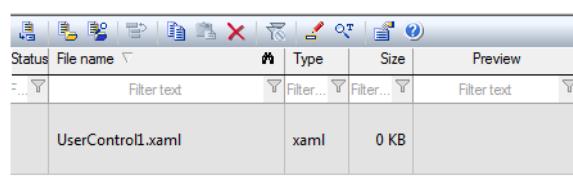
In the next step, how the DLL and XAML file are incorporated into zenon is explained.

## STEPS IN ZENON

1. Open the zenon Editor
2. Go to *File -> Graphics*.
3. Select **Add file...** in the context menu



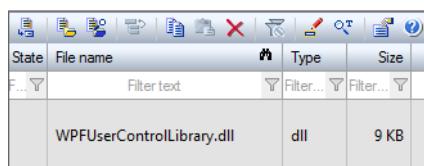
4. Select the XAML file at the save location (**UserControl1.xaml** from the **WPFUserControlNoCodeBehind** project) and add this:



5. Insert the DLL with the functionality for the XAML file.

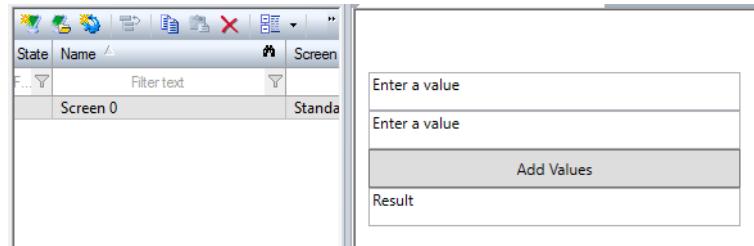
To do this:

- a) Select, in the context menu, File -> Other**Add file....**
- b) Select the file **WPFUserControlLibrary.dll** (from the output path) of the first project (**WPFUserControlLibrary**).

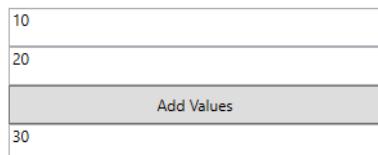


6. Create a zenon screen.
7. Add a WPF element and select the previously-incorporated XAML file.

You should now see the following in the zenon Editor:



8. Start zenon Runtime in order to also test the control there.



## Information

The XAML file and referenced assemblies can also be saved in complied form as a \*.cdwpf file. Only one file thus need to be imported in the Editor (under *Files -> Graphics*). Further information on this can be found in the CDWPF files (collective files) (à la page 48) chapter.

**Hint:** When developing a WPF user control, it is usually more practical to insert the XAML file and the referenced DLL(s) separately. This makes the replacement of the DLL and debugging easier. Further information on the topic of debugging can be found in the Debugging the WPF user control in the Runtime (à la page 29) chapter.

## thumb-up Hint

DLLs that are part of a WPF element can also be replaced during ongoing operation. In doing so, the referencing is via linking in the XAML file.

To replace a DLL:

- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In Explorer, replace the DLL in the `\wpfcache` folder of the Editor files.  
You can find this folder in the SQL directory under  
`...\\PROJECT-GUID\\FILES\\zenon\\custom\\wpfcache`

As an alternative to replacement using Explorer, you can also replace the file in the zenon Editor directly. To do this, carry out the following steps:

- ▶ In the Visual Studio project settings, increase the file version of the DLL.
- ▶ Create the new DLL.
- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In the zenon Editor, delete the DLL from the `\Files\Other` folder and add the file with the higher version number.

Further examples can be found in the Examples: Integration of WPF into zenon (à la page 114) chapter.

## 5.2 Debugging the WPF user control in the Runtime

To debug the WPF user control in the Runtime, proceed as follows.

In this example, the control described in the Creation of a simple WPF user controls with code behind function (à la page 24) is used.

### DEBUGGING BY MEANS OF ATTACH TO PROCESS

1. Ensure that zenon Runtime has been started and a screen with the WPF user control is open.  
Furthermore, ensure that the DLL that is currently being used corresponds to the build (Version) of the user control project (**WPFUserControlLibrary**).

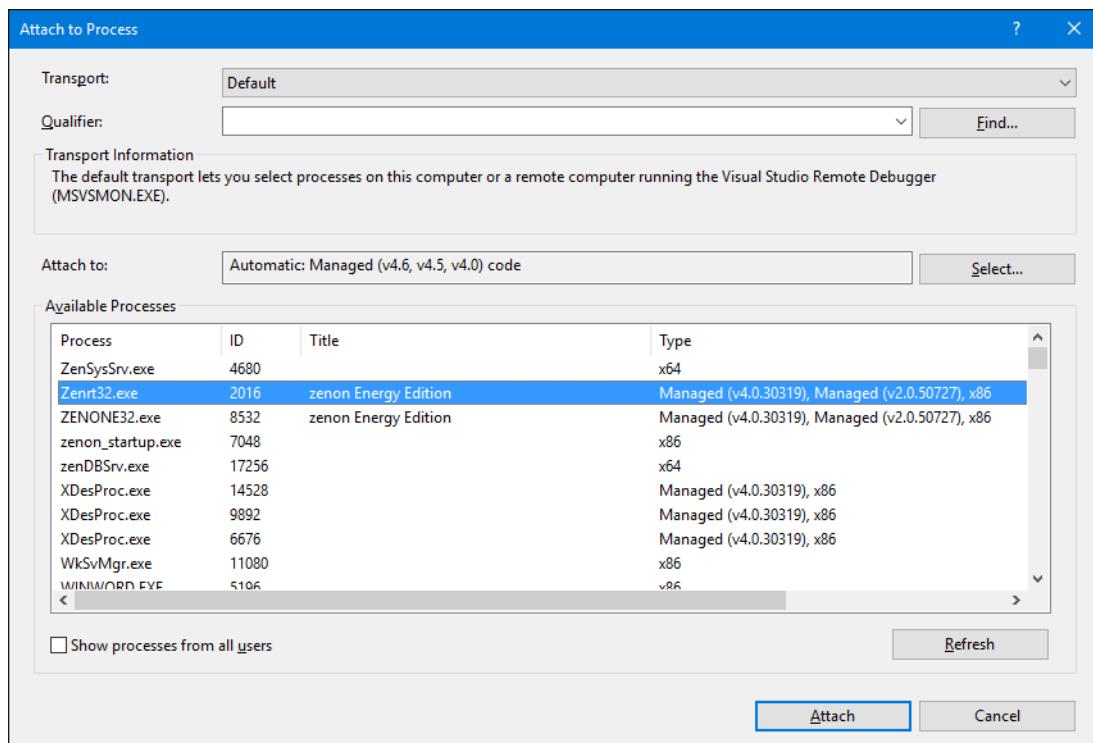
2. Set a breakpoint in the click event of the button in the Visual Studio project

```

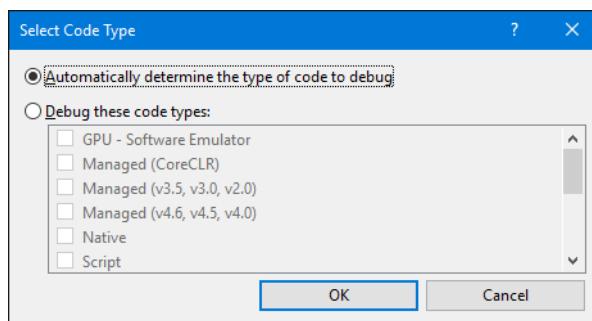
28 private void buttonAdd_Click(object sender, RoutedEventArgs e)
29 {
30 try
31 {
32 textBoxC.Text = (Convert.ToInt32(textBoxA.Text) + Convert.ToInt32(textBoxB.Text)).ToString();
33 }
34 catch (Exception ex)
35 {
36 textBoxC.Text = "Error adding values: " + ex.Message;
37 }
38 }

```

3. In Visual Studio, under **Debug**, select the **Attach to Process** menu item.
4. Select the zenon Runtime process

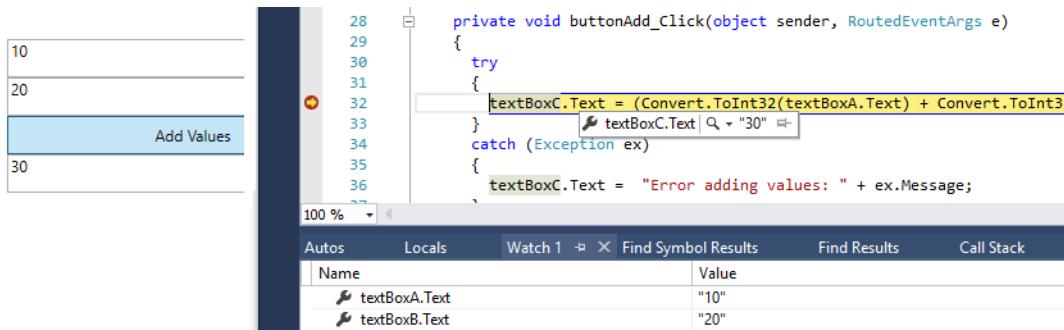


5. Under **Attach to**, select either **Automatic** or the corresponding .NET framework version (**v4.x** in this case)



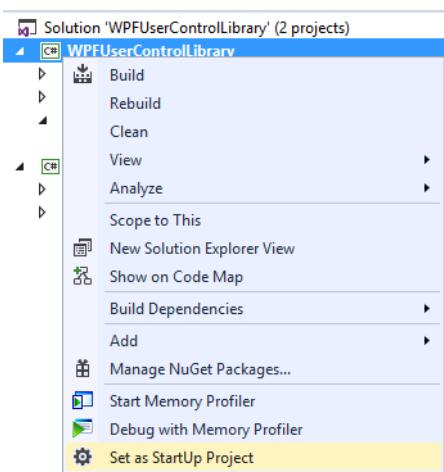
6. Click on **Attach**

- Now trigger the breakpoint in which you enter values into the WPF control in zenon Runtime and click on the button



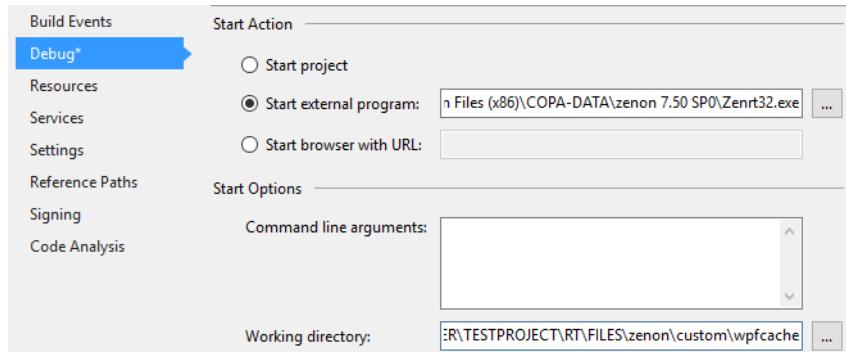
## DEBUG USING START EXTERNAL PROGRAM

- Ensure that zenon Runtime has been closed.
- Ensure that, in the zenon Editor, the project that contains the WPF user control has been set as the start project.
- Ensure that the user control project (**WPFUserControlLibrary**) is set as the start project in Visual Studio.

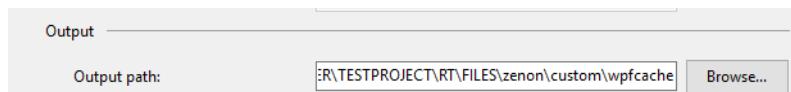


- In the project properties of the Visual Studio project, select under **Debug**, for **Start action: Start external program**
- For **Start external program**, select the path of the zenon Runtime application.
- Under Working Directory, select the **\wpfcache** folder of the Runtime files  
(...\\PROJECTNAME\\RT\\FILES\\zenon\\custom\\wpfcache)

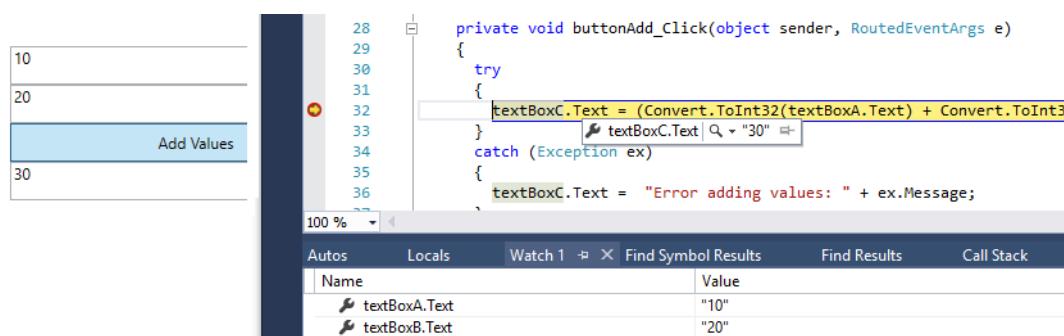
**Hint:** In the selected project in the zenon Editor, press the keyboard combination **CTRL+ALT+R** in order to jump directly to the root directory of the Runtime files.



7. In the project properties, enter \wpfcache folder of the Runtime files as the **Output path** under **Build**.



8. Create the project in Visual Studio
9. Start debugging in Visual Studio with **Start**
10. zenon Runtime is now started automatically.
11. Trigger the breakpoint by entering values in the WPF control in zenon Runtime and click on the button



## Information

When starting zenon Runtime, the assemblies (DLLs) referenced in the WPF user controls from the **\FILES\zenon\custom\additional** folder, and/or the assemblies from **CDWPF** files in the **\FILES\zenon\custom\wpfcache** folder are copied. If the file version of the DLL in the **\wpfcache** folder is one higher than the version of the "original file", it is not replaced!

For debugging, it is thus sufficient to only replace the file that is on the **\wpfcache** folder directly.

For delivery, it must be ensured that the current version of the DLL is present in the **\additional** folder or the **CDWPF** file!

**Attention:** If only the DLL is updated in the **\additional** folder or in the **CDWPF**, but the version number is not increased, the DLL must be deleted manually in the **\wpfcache** folder, because it is not updated otherwise (due to the above-described mechanism).

## Hint

DLLs that are part of a WPF element can also be replaced during ongoing operation. In doing so, the referencing is via linking in the XAML file.

To replace a DLL:

- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In Explorer, replace the DLL in the **\wpfcache** folder of the Editor files.  
You can find this folder in the SQL directory under  
**...\\PROJECT-GUID\\FILES\\zenon\\custom\\wpfcache**

As an alternative to replacement using Explorer, you can also replace the file in the zenon Editor directly. To do this, carry out the following steps:

- ▶ In the Visual Studio project settings, increase the file version of the DLL.
- ▶ Create the new DLL.
- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In the zenon Editor, delete the DLL from the **\Files\Other** folder and add the file with the higher version number.

▶

## 5.3 Data exchange between zenon and WPF user controls

There are different possibilities for exchanging data between zenon and WPF user controls.

- ▶ Data exchange using dependency properties (à la page 34)
- ▶ Data replacement via VSTA (à la page 38)

### 5.3.1 Data exchange using dependency properties

The most elegant and secure way to exchange data between zenon and self-created WPF user controls is by using *Dependency Properties*.

The WPF user control project created in the Creating a simple WPF user controls with code behind function (à la page 24) serves as a basis (**WPFUserControlLibrary**).

In this chapter, the focus is purely on the core theme (*Dependency Properties* and data exchange between the user control and zenon in this case). Specific WPF features such as Databinding, etc., as well as explicit error handling, are not covered.

#### ADDITIONS TO THE CODE

1. Create the *TextChanged* Event for the **textBoxA** element in the file **UserControl1.xaml**

**TextChanged="textBoxA\_TextChanged"**

2. Add the following lines of code in the **UserControl1** class of the code behind file (**UserControl1.xaml.cs**)

```
/// <summary>
/// Gets or sets the ValueA.
/// </summary>
public double ValueA
{
 get
 {
 return (double)GetValue(ValueADependencyProperty);
 }
 set
 {
 SetValue(ValueADependencyProperty, value);
 }
}
```

```
/// <summary>
/// Dependency property for ValueA
/// </summary>
public static readonly DependencyProperty ValueADependencyProperty =
DependencyProperty.Register("ValueA", typeof(double),
typeof(UserControl1), new FrameworkPropertyMetadata(0.0, new
PropertyChangedCallback(OnValueADependencyPropertyChanged)));

/// <summary>
/// Called when [value a dependency property changed].
/// </summary>
/// <param name="source">The source.</param>
/// <param name="e">The <see cref="DependencyPropertyChangedEventArgs"/> instance containing the
event data.</param>
private static void OnValueADependencyPropertyChanged(DependencyObject source,
DependencyPropertyChangedEventArgs e)
{
 UserControl1 control = source as UserControl1;
 if (control != null)
 {
 try
 {
 control.ValueA = (double)e.NewValue;
 control.textBoxA.Text = control.ValueA.ToString();
 }
 catch (Exception)
 {}
 }
}

/// <summary>
/// Handles the TextChanged event of the textBoxA control.
/// </summary>
/// <param name="sender">The source of the event.</param>
/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event
data.</param>
private void textBoxA_TextChanged(object sender, TextChangedEventArgs e)
{
 try
 {
```

```
ValueA = Convert.ToDouble(textBoxA.Text);
}
catch (Exception)
{}
}
```

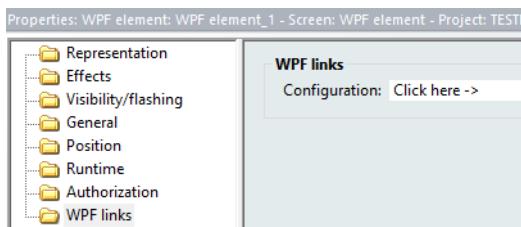
Then build the solution.

### Information

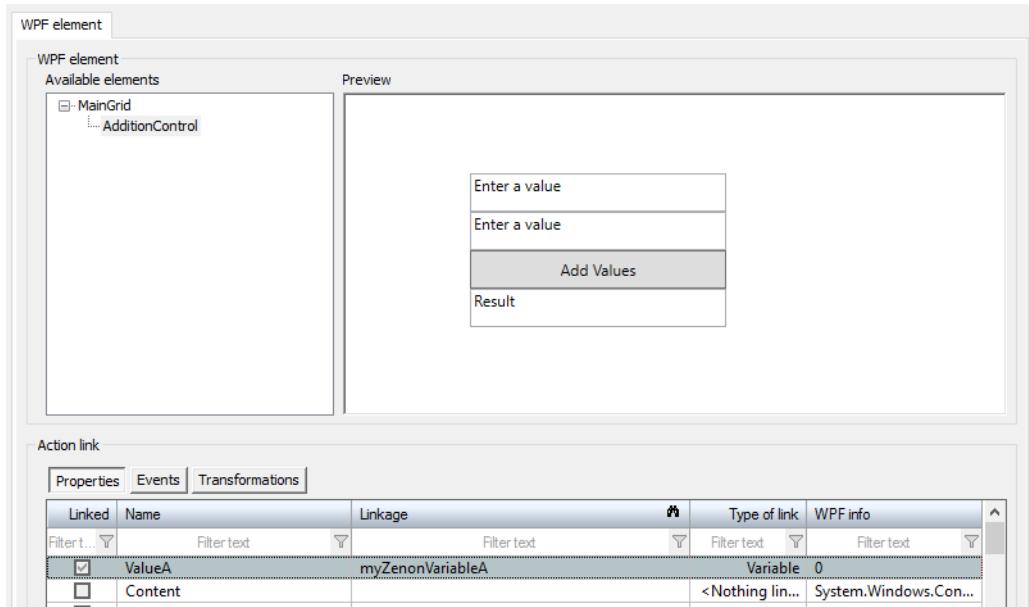
A numerical property (**double**) is used in this example. Other simple data types (such as **bool**, **string**, **int**, etc.) can also be used.

## LINKING IN ZENON

1. Update the WPF user control (DLL) in the zenon Editor.
2. Proceed as described in the creation of a simple WPF user controls with code behind function (à la page 24) chapter.
3. Create a numeric variable in zenon. Link this variable to a dynamic text element. You place the dynamic text element in the screen next to the WPF element with your user control.
4. Open the screen that contains the WPF element and select, for the WPF element, under **WPF links: Configuration**



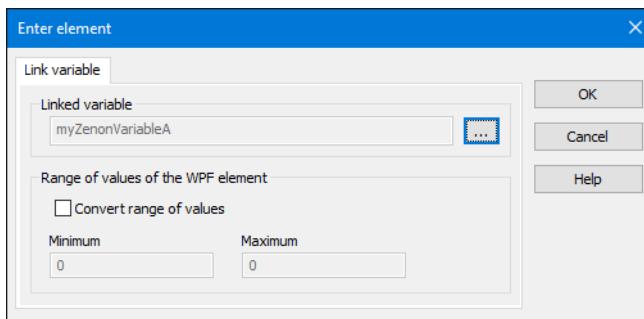
5. Expand the node in the tree at the top left and select **AdditionControl**



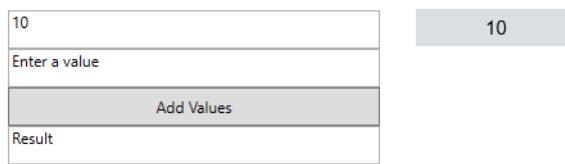
6. Select the line with *ValueA* (this is the name of the property that was created in the code beforehand) and select, for **Type of link: Variable**

**Hint:** Give Properties a prefix so that this can be found more easily, for example: *\_ValueA*

7. In the column under **Linkage**, print out the variable that was created in zenon beforehand



8. Confirm the dialog with OK and build the Runtime files
9. Start Runtime in order to test the WPF user control



10. If the value is changed in user control, the value automatically changes in zenon and vice versa.
11. You can of course debug the control as described in the Debugging the WPF user control in Runtime (à la page 29) chapter, and create further dependency properties.

## Information

The [UserControl\\_Loaded](#) event can be used in order to (automatically) access the values of the dependency property during initialization (when calling up the user control) for example.

### 5.3.2 Data replacement via VSTA

Data can also be exchanged between zenon and WPF user controls using VSTA.

The API element methods

- ▶ [get\\_WPFPProperty](#) (reading of values)
- ▶ [set\\_WPFPProperty](#) (writing of values)

are used for this.

The example used here is based on the example used in the Data exchange using dependency properties (à la page 34) chapter.

### CREATION OF A VSTA MACRO FOR DATA EXCHANGE BETWEEN ZENON AND THE WPF USER CONTROL

1. Create the following VSTA macro in the project add-in of the zenon project

```
/// <summary>
/// Sample Macro for data exchange between VSTA and a WPF User Control
/// </summary>
public void MacroWPFAccess()
{
 //Get the Screen and Element hosting the WPF User Control
 zenOn.IDynPicture myWPFScreen = this.DynPictures().Item("Screen");
 zenOn.IElement myWPFElement = myWPFScreen.Elements().Item("WPF_Element");

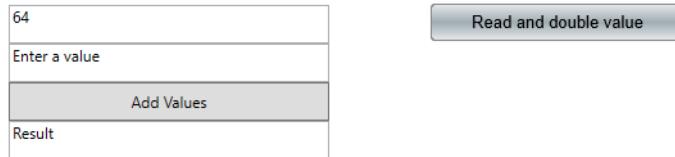
 //Read the current value from the WPF Element property
 double currentValue = Convert.ToDouble(myWPFElement.get_WPFPProperty("AdditionControl", "ValueA"));

 //Double the value and write it back to the WPF Element property
 myWPFElement.set_WPFPProperty("AdditionControl", "ValueA", currentValue * 2);
}
```

Whereby:

- ▶ "**Screen**" is the name of the zenon screen in which the WPF element is located
- ▶ "**WPF\_Element**" is the name of the WPF element that contains the WPF user control
- ▶ "**AdditionControl**" is the name of the WPF user controls itself (defined in the **UserControl1.xaml** file)
- ▶ "**ValueA**" is the name of the user control property

2. Create an execute VSTA macro function and link this to a button in the screen in which the WPF element is also located
3. Start Runtime to test changes



When executing the macro, the value is read by the control, doubled and written back.

#### Information

The user control properties used for this method of data exchange need not necessarily be dependency properties, as outlined in this example. "Standard" properties can also be used, see in relation to this the Access via VSTA "variable link" (à la page 39) chapter.

## 5.4 Access to the zenon (Runtime) object model from a WPF user control

There are different possibilities for access to the zenon object model from a WPF user control. This is explained in more detail in the following chapters.

#### Attention

When using zenon COM objects with self-created user controls or external applications, they must be enabled using the **Marshal.ReleaseComObject** method. Enabling by means of the **Marshal.FinalReleaseComObject** method must not be used, because this leads to a malfunction of zenon Add-ins.

### 5.4.1 Access via VSTA "variable link"

In order to get access to the zenon Runtime COM interface by means of "variable link", proceed as follows. The creation of a simple WPF user controls with code behind function (à la page 24) serves as an initial example.

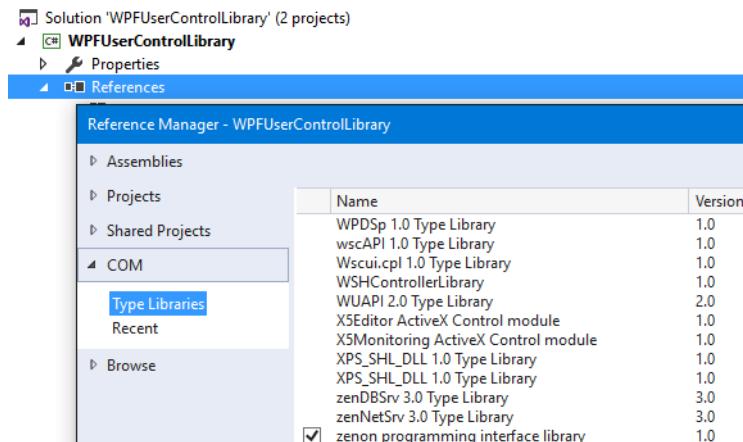
## Information

The following code is intended to show an example of how the COM implements access to zenon Runtime and in doing so limits itself to the basic functionality. There is no explicit error handling, etc.

## NECESSARY AMENDMENTS IN WPF USER CONTROL

The following steps are necessary in the WPF user control project (**WPFUserControlLibrary**).

Firstly, a reference to the zenon COM interface must be incorporated.



After this, the following code must be inserted in the **UserControl1** class:

```
//The zenon Project
zenOn.Project zenonProject = null;

/// <summary>
/// Property for the Variable link via VSTA
/// </summary>
public object zenonVariableLink
{
 get { return null; }
 set
 {
 if (value != null && zenonProject == null)
 {
 zenOn.Variable zenonVariable;
```

```
try
{
 zenonVariable = (zenOn.Variable)value;
}
catch (Exception)
{
 return;
}

if ((zenonVariable!= null) && (!string.IsNullOrEmpty(zenonVariable.Name)))
{
 zenonProject = zenonVariable.Parent.Parent;
}
}
}

/// <summary>
/// Trigger used to notify the control from VSTA to release the COM resources
/// </summary>
public object zenonReleaseTrigger
{
 get { return null; }
 set
 {
 if ((bool)value && zenonProject != null)
 {
 try
 {
 Marshal.ReleaseComObject(zenonProject);
 }
 catch (Exception)
 {
 return;
 }
 zenonProject = null;
 GC.Collect();
 GC.WaitForPendingFinalizers();
 GC.Collect();
 }
 }
}
```

```
}
```

```
}
```

Whereby access to the properties `zenonVariableLink` (to initialize the COM object) and `zenonReleaseTrigger` (to unlock the COM object) are subsequently accessed from VSTA (write).

In order to test the COM access quickly very easily, it is possible to insert the following line of code in the existing button click event of the user control.

```
private void buttonAdd_Click(object sender, RoutedEventArgs e)
{
 if (zenonProject != null)
 {
 MessageBox.Show(zenonProject.Name);
 }
 return;
...
}
```



## Information

A `zenOn.Project` variable is used in this example. Of course other objects such as events, etc. of the zenon object model can also be used.

## NECESSARY AMENDMENTS IN THE ZENON PROJECT/VSTA CODE

The following steps are necessary in the VSTA code:

Creation of a VSTA macro for the initialization

```
/// <summary>
/// Macro for API initialization in the WPF User Control
/// </summary>
public void MacroWPFInit()
{
 zenOn.IDynPicture myWPFScreen = this.DynPictures().Item("Screen");
 zenOn.IElement myWPFElement = myWPFScreen.Elements().Item("WPF_Element");
 myWPFElement.set_WPFPProperty("AdditionControl", "zenonVariableLink", this.Variables().Item(0));
}
```

Creation of a VSTA macro for approval

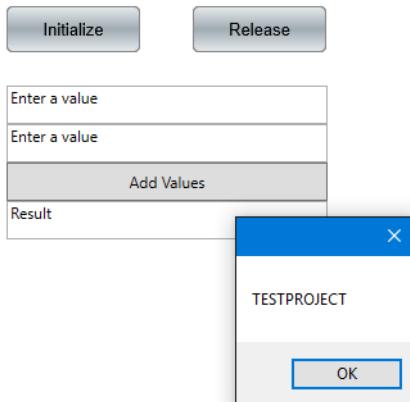
```
/// <summary>
```

```
/// Macro for API release in the WPF User Control
/// </summary>
public void MacroWPFRlease()
{
 zenOn.IDynPicture myWPFScreen = this.DynPictures().Item("Screen");
 zenOn.IElement myWPFElement = myWPFScreen.Elements().Item("WPF_Element");
 myWPFElement.set_WPFProperty("AdditionControl", "zenonReleaseTrigger", true);
}
```

Create two execute VSTA macro functions that are linked with buttons, which are in the same screen as the WPF element.

Now start Runtime in order to test the functionality

- ▶ Execute the macro for initialization
- ▶ Click on the button in the WPF user control; a message box with the project name of the project appears



- ▶ Execute the macro for release

In order to debug the user control, it is possible to proceed as described in the Debugging the WPF user control in Runtime (à la page 29).

### Hint |

The initialization and release of the COM object in this example is only carried out for simple demonstration using VSTA macro functions. Depending on the application, and/or in practice, events in VSTA are better suited to this.

For example, the code for initialization in the [\\_Open](#) event of the screen can be executed with the WPF element and the code for release in the [\\_Close](#) event.

The mechanism described here is also used in the Display of WPF elements in the zenon Web Client (à la page 110) chapter.

### ⚠ Attention

If COM objects are used in WPF user controls, these must always be explicitly approved before destroying the WPF user control (before closing the screen, before closing Runtime, before reloading).

## 5.4.2 Access via marshaling

In order to get access to the zenon Runtime COM interface by means of marshaling, proceed as follows. The creation of a simple WPF user controls with code behind function (à la page 24) serves as an initial example.

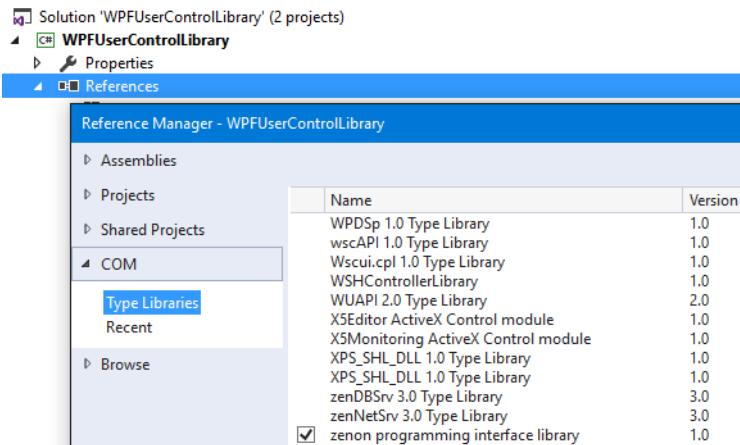
### 💡 Information

The following code is intended to show an example of how the COM implements access to zenon Runtime and in doing so limits itself to the basic functionality. There is no explicit error handling, etc.

## NECESSARY AMENDMENTS IN WPF USER CONTROL

The following steps are necessary in the WPF user control project ([WPFUserControlLibrary](#)).

Firstly, a reference to the zenon COM interface must be incorporated.



After this, the following code must be inserted in the `UserControl1` class:

```
//The zenon Project
zenOn.Project zenonProject = null;
```

Furthermore, the constructor of the user controls must be supplemented with the lines below (to initialize the COM object):

```
/// <summary>
/// Constructor for UserControl1, initialize COM Object
/// </summary>
public UserControl1()
{
 InitializeComponent();

 try
 {
 zenonProject =
 ((zenOn.Application)Marshal.GetActiveObject("zenOn.Application")).Projects().Item("TESTPROJECT");
 }
 catch (Exception)
 {
 }
}
```

The COM object must be approved in the `UserControl_Unloaded` event:

```
/// <summary>
/// Release COM Object
/// </summary>
private void UserControl_Unloaded(object sender, RoutedEventArgs e)
{
try
{
if (zenonProject != null)
{
 Marshal.ReleaseComObject(zenonProject);
 zenonProject = null;
}
}
catch (Exception)
{
}
}
```

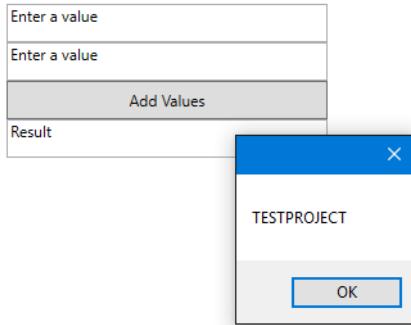
In order to test the COM access quickly very easily, it is possible to insert the following line of code in the existing button click event of the user control.

```
private void buttonAdd_Click(object sender, RoutedEventArgs e)
{
if (zenonProject != null)
{
 MessageBox.Show(zenonProject.Name);
}
return;
```

...

Now build the solution and update the WPF user control in the zenon project.

Start Runtime to test the user control.



In order to debug the user control, it is possible to proceed as described in the Debugging the WPF user control in Runtime (à la page 29).

### Information

A [zenOn.Project](#) variable is used in this example. Of course other objects such as events, etc. of the zenon object model can also be used.

### Attention

If COM objects are used in WPF user controls, these must always be explicitly approved before destroying the WPF user control (before closing the screen, before closing Runtime, before reloading).

### Information

No access by means of marshaling is possible in the zenon web client. If access to the COM interface is required there, the method described in the Access via VSTA "variable link" (à la page 39) must be used.

## 6 Engineering in zenon

In order to be able to use WPF user controls in zenon, version 3.5 (or higher, depending on the .NET framework version used in the user control) of the Microsoft framework must be used on both the Editor computer and the Runtime computer.

## CONDITIONS FOR THE WPF DISPLAY IN ZENON

The dynamization is currently available for simple variable types (numerical data types as well as string). Arrays and structures cannot be dynamized.

Therefore the following WPF functions can be implemented in zenon:

- ▶ Element properties that correspond to simple data types, such as *String*, *Int*, *Double*, *Bool* etc.
- ▶ Element properties of the "Object" type, which can be set with simple data types
- ▶ Element events can be used with functions; the parameters of the events are not however available in and cannot be evaluated in zenon
- ▶ Element transformation, for which a **RenderTransform** is present for the element in the XAML file

**Attention:** if the content is outside of the area of the WPF element during transformation, this is not labeled

**Notes on dBase:** No shade can be displayed in zenon for WPF elements.

### Attention

If the Runtime files were created for a project for a version before 6.50, existing **WPF elements** are not included into Runtime screens.

## 6.1 CDWPF files (collective files)

A CDWPF file (with the suffix **\*.cdwpf**) is a renamed ZIP file that contains the following components:

- ▶ XAML file (to reference the user control assembly)
- ▶ DLL file (the actual WPF user control, optional)
- ▶ Preview graphics (for preview, optional)

Rules for the use of collective files:

- ▶ The files (XAML, DLL, preview graphics) can be in the CDWPF file directly or in a joint folder.
- ▶ The name of the collective file should correspond to the names of the XAML file.
- ▶ Only one XAML file may be contained.
- ▶ The preview graphic should be small and no more than 64 pixels high.  
Name of the preview file: **preview.png** or the name of the XAML file with the suffix **.png**.
- ▶ Any number of assemblies can be used. The distinction is made on the basis of the file version.
- ▶ Collective files do not need to contain an assembly.
- ▶ All subfolders are examined and only taken into account with **\*.dll**, **\*.xaml** or **\*.png** files.

- ▶ If a collective file (\*.cdwfp) is replaced by a file with a different version, all corresponding CDWPF files in all symbols and images in all projects must be adapted.

## 6.2 create WPF element

To create a WPF element

1. In the elements toolbar, select the symbol for **WPF element** or the **Elements** entry in the menu
2. Select the start point in the main window.
3. Pull open the element with the mouse.
4. In properties, select **Affichage** the property **Fichier XAML** in the group.
5. The file selection dialog opens.
6. Select the desired file  
Files of the following formats are valid:
  - ▶ \*.xaml: Extensible Application Markup Language
  - ▶ \*.cdwfp: WPF collective file, also shows preview image(The file must already be present in the Project Manager under **Files/graphics** or created in the dialog.)
7. Configure the links (à la page 49).



### Information

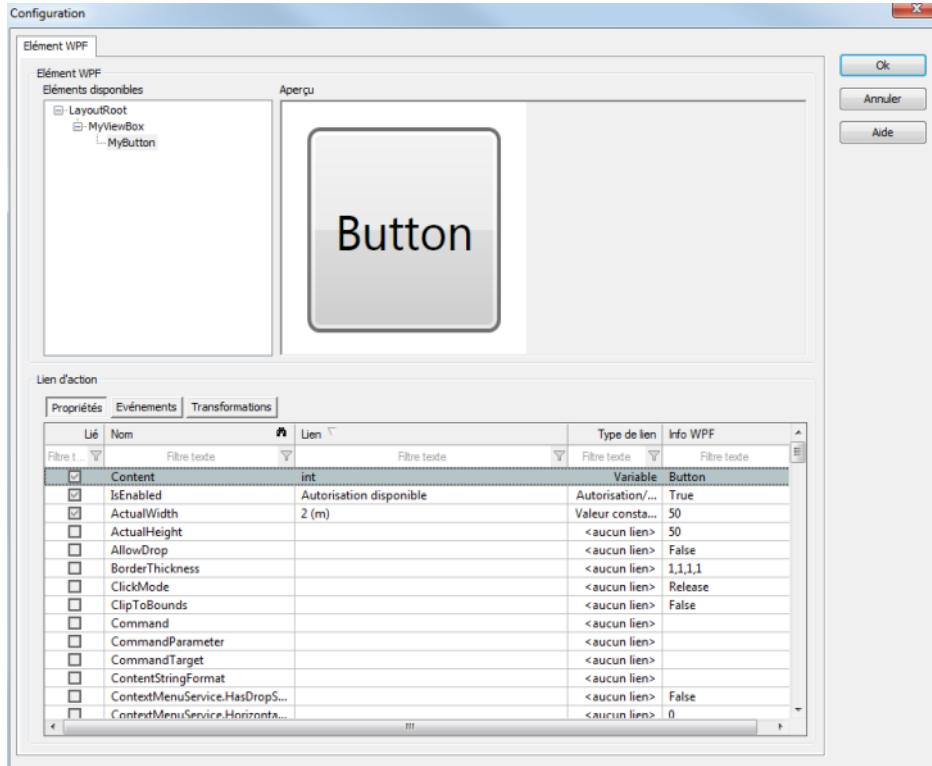
If referenced assemblies are used, note the instructions in the Referenced assemblies (à la page 8) chapter.

## 6.3 Configuration of the linking

To configure a WPF element

1. In properties, select **Liens WPF** the property **Configuration** in the group.
2. The dialog with three tabs opens with a preview of the XAML file and the elements present in the file

## DIALOG CONFIGURATION



| Parameter                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Available elements</b>        | <p>Shows the named file elements in a tree structure. The selected element can be linked with process data.</p> <p><b>WPF</b> is assigned to process data based on the element name. Therefore elements are only shown if they and the attendant elements have a name. Allocations are configured and shown in the <b>Properties</b>, <b>Events</b>, <b>Transformations</b> tabs.</p> <p><b>Hint:</b> If the corresponding elements are not displayed, check whether the non-displayed objects in the XAML file have been given a name (for example: &lt;Grid Name="GridName"&gt;).</p> |
| <b>Preview</b>                   | The selected element is shown flashing in the preview.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Properties</b> (à la page 51) | Configuration and display of properties (variables, authorizations, interlockings, linked values).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Events</b> (à la page 58)     | Configuration and display of events (functions).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Parameter                             | Description                                                                                                                                      |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Transformations</b> (à la page 59) | Configuration and display of transformations.                                                                                                    |
| <b>Name</b>                           | Name of the property.                                                                                                                            |
| <b>Linkage</b>                        | Selection of link.                                                                                                                               |
| <b>Type of link</b>                   | Type of link (variable, authorization, function)                                                                                                 |
| <b>WPF info</b>                       | Shows the current value for properties in WPF content. For the user, it is directly visible what type of property it is (Boolean, string, etc.). |
| <b>Linked</b>                         | Shows if a property is currently being used.<br>If not contained in the view, can be selected via <b>Context menu-&gt;Column selection</b> .     |



## Information

Only logical objects can be displayed in the configuration dialog. Visual objects are not displayed. You can read background information and how visual objects can be animated in the Allocation of zenon object to WPF content.

### EDIT HYPERLINKS

All configured hyperlinks can be edited from the properties of the element. Click on the element and open the property group **Liens WPF**. Hyperlinks can be further configured here, without having to open the dialog.

Limitations :

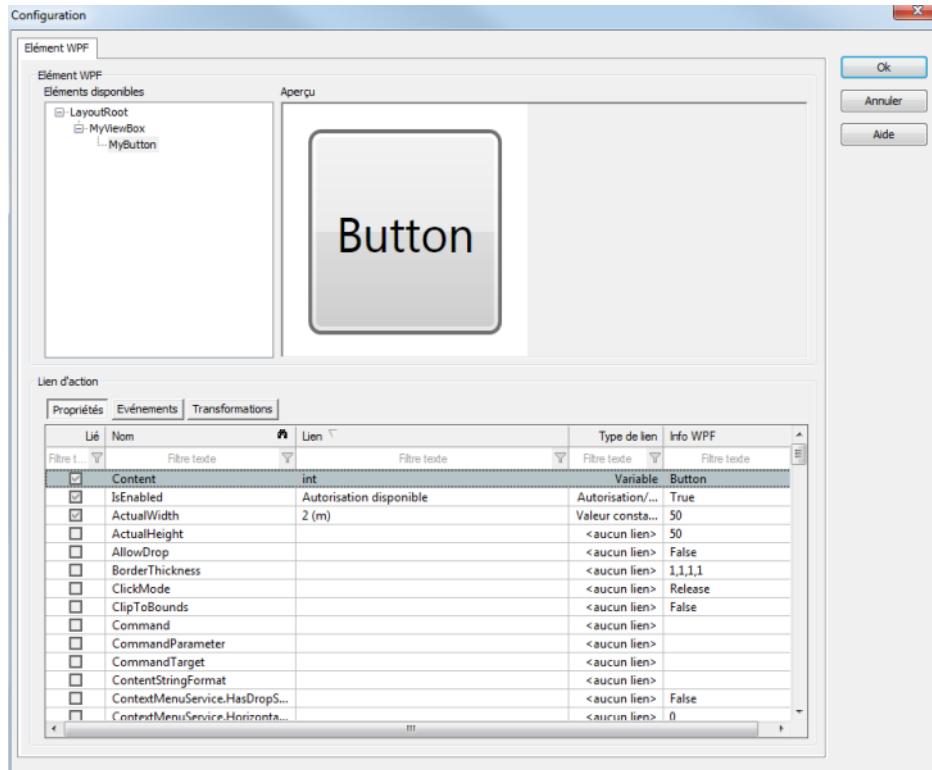
- ▶ Le type de liaison ne peut pas être modifié ici.
- ▶ Les nouveaux liens peuvent uniquement être créés via la boîte de dialogue de configuration.
- ▶ Insertion d'un élément WPF dans un symbole : Les liens WPF ne peuvent pas être exportés.

### 6.3.1 Properties

The properties enable the linking of:

- ▶ Variables (à la page 53)
- ▶ Values (à la page 55)

- ▶ Authorizations and interlockings (à la page 56)



| Parameter    | Description                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name         | Name of the property.                                                                                                                                                   |
| Lien         | Linked variable, authorization or linked value.<br>Clicking in the column opens the respective selection dialog, depending on the entry in the <b>Link type</b> column. |
| Type de lien | Selection of linking.                                                                                                                                                   |
| Info WPF     | Shows the current value for properties in WPF content. For the user, it is directly visible what type of property it is (Boolean, string, etc.).                        |
| Lié          | Shows if a property is currently being used.<br>If not contained in the view, can be selected via <b>Context menu-&gt;Column selection</b> .                            |

## CREATE LINK

To create a link:

1. Highlight the line with the property that is to be linked

2. Click in the **Link type cell**

3. Select the desired link from the drop-down list.

The following are available:

- ▶ <not linked> (deletes an existing link)
- ▶ Authorization/Interlocking
- ▶ Constant value
- ▶ Variable

4. Click in the **Link** cell

5. The dialog for configuring the desired link opens

### Information

Properties of WPF and zenon can be different. If, for example the **visibility** property is linked, there are three values available in .NET:

- ▶ 0 - visible
- ▶ 1 - invisible
- ▶ 2- collapsed

These values must be displayed via the linked zenon variable.

#### 6.3.1.1 Link variable

To link a variable with a WPF property:

1. Highlight the line with the property that is to be linked

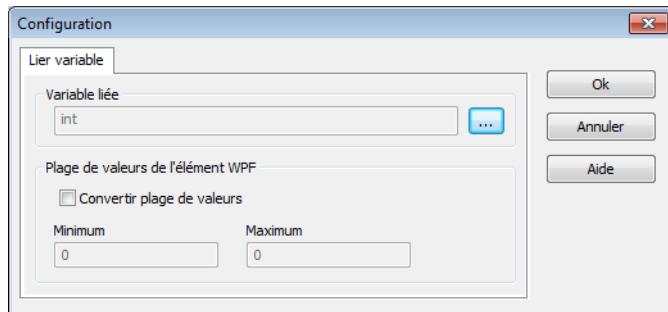
2. Click in the **Link type cell**

3. Select from the **variable** drop down list

4. Click in the **Link** cell

5. The dialog for configuring the variables opens

This dialog also applies for the selection of variables with transformations (à la page 59). The configuration also makes it possible to convert from zenon into WPF units.



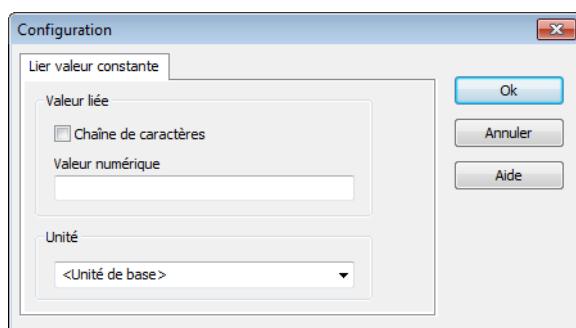
| Parameter                                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Linked variable</b>                    | Selection of the variable to be linked. A click on the ... button opens the selection dialog.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Range of values of the WPF element</b> | Data to convert variable values into WPF values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Convert range of values</b>            | <p>Active: WPF unit conversion is switched on.</p> <p><b>Effect in the Runtime:</b> The current zenon value (incl. zenon unit) is converted to the WPF range using standardized minimum and maximum values.</p> <p><b>For example:</b> The value of a variable varies from 100 to 200. With the variables, the standardized range is set to 100 - 200. The aim is to display this change in value using a WPF rotary knob. For this:</p> <ul style="list-style-type: none"> <li>▶ for <b>Transformations</b>, the <b>RotateTransform.Angle</b> property is linked to the variables</li> <li>▶ <b>Adjust value</b> activated</li> <li>▶ a WPF value range of 0 to 360 is configured</li> </ul> <p>Now the rotary knob can be turned at a value of 150, for example, by 180 degrees.</p> |
| <b>Minimum</b>                            | Defines the lowest WPF value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Maximum</b>                            | Defines the highest WPF value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>OK</b>                                 | Accepts settings and ends the dialog.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Cancel</b>                             | Discards settings and ends the dialog.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Help</b>                               | Opens online help.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

### 6.3.1.2 Link values

Linked values can either be a **String** or a numerical value of the type **Double**. When selecting the screen, the selected value is sent in WPF content after loading the WPF content.

To link a value to a WPF property:

1. Highlight the line with the property that is to be linked
2. Click in the **Link type cell**
3. Select **Value linkings** from the drop-down list
4. Click in the **Link** cell
5. The dialog for configuration of value linking opens



| Parameter                           | Description                                                                                                                                                                                                                                                                                                                      |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Linked value:</b>                | Entry of a numerical value or string value.                                                                                                                                                                                                                                                                                      |
| <b>Use string</b>                   | <p><b>Active:</b> A string value is used instead of a numerical value.</p> <p>The language of string values can be switched. The text is translated in the Runtime when the screen is called up and sent in WPF content. If the language is switched whilst the screen is opened, the string value is retranslated and sent.</p> |
| <b>String value/numerical value</b> | Depending on what is selected for the <b>Use string</b> property, a numerical value or a string value is entered into this field. A unit of measurement can also be selected for numerical values.                                                                                                                               |
| <b>Unit:</b>                        | <p>Selection of a unit of measurement from the drop down list. You must have configured this in unit switching beforehand.</p> <p>The unit of measurement is allocated with the numerical value. If the units are switched in the Runtime, the value is converted to the new unit of measurement and sent to WPF content.</p>    |

## CLOSE DIALOG

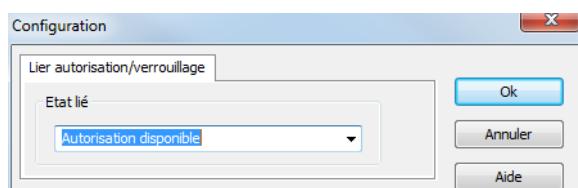
| Options        | Description                                                    |
|----------------|----------------------------------------------------------------|
| <b>OK</b>      | Applique les paramètres et ferme la boîte de dialogue.         |
| <b>Annuler</b> | Annule toutes les modifications et ferme la boîte de dialogue. |
| <b>Aide</b>    | Opens online help.                                             |

### 6.3.1.3 Link authorization or interlocking

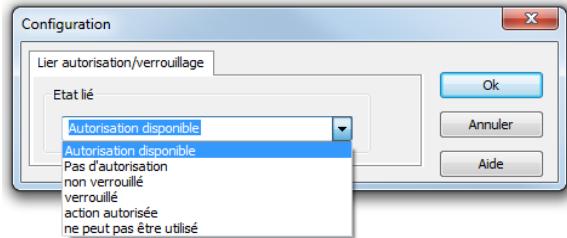
Authorizations cannot be granted for the whole WPF element. The element is allocated a user level. Authorizations are granted within the user level for individual controls. If an authorization is active, the value 1 is written to the element.

To link a user authorization or interlocking to a WPF property:

1. Highlight the line with the property that is to be linked
2. Click in the **Link type cell**
3. Select **Authorization/interlocking** from the drop down menu
4. Click in the **Link** cell
5. The dialog for configuring the authorizations opens



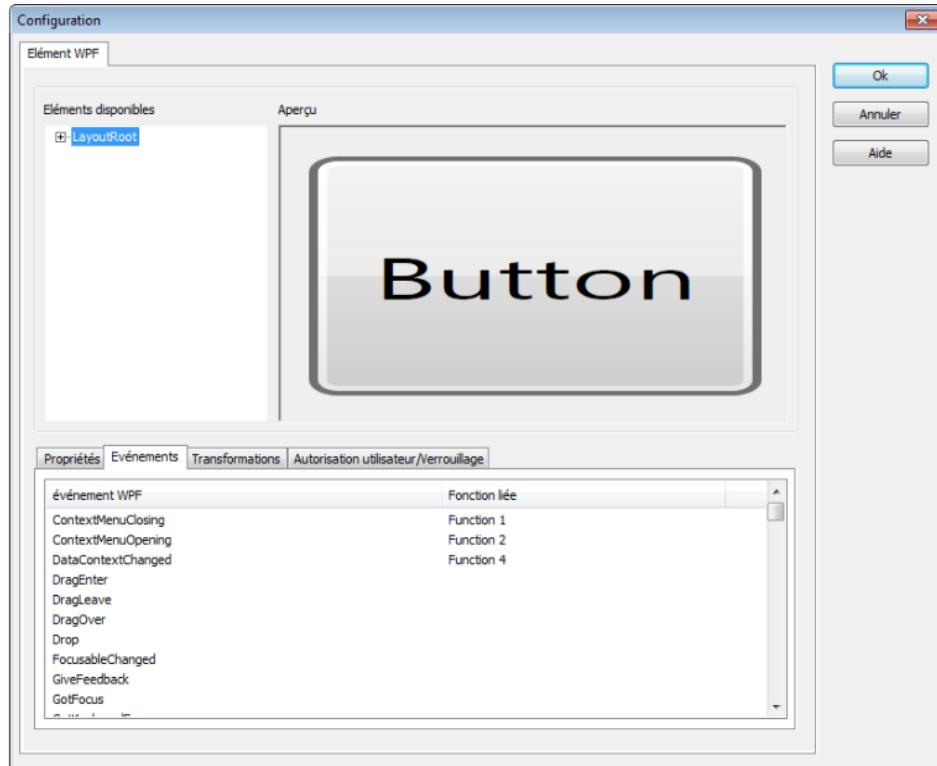
| Parameter                              | Description                                                                                                                                                                   |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Link authorization/interlocking</b> | Setting the authorizations.                                                                                                                                                   |
| <b>Linked status</b>                   | Selection of an authorization that is linked to a WPF control from the drop down list. For example, visibility and operability of a WPF button can depend on a user's status. |



| Authorization                       | Description                                                                                                              |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>Authorization granted</i>        | If the user has sufficient rights to operate the <b>WPF element</b> , a value of 1 is written to the property.           |
| <i>Authorization does not exist</i> | If the user does not have sufficient rights to operate the <b>WPF element</b> , a value of 1 is written to the property. |
| <i>Not interlocked</i>              | If the element is not locked, the value 1 is written to the property.                                                    |
| <i>Interlocked</i>                  | If the element is locked, the value 1 is written to the property.                                                        |
| <i>Can be operated</i>              | If authorization is present and the element is not locked, then a value of 1 is written to the property.                 |
| <i>Cannot be operated</i>           | If authorization is not present or the element is not locked, then a value of 1 is written to the property.              |

### 6.3.2 Events

Events make it possible to link zenon functions to a WPF element.

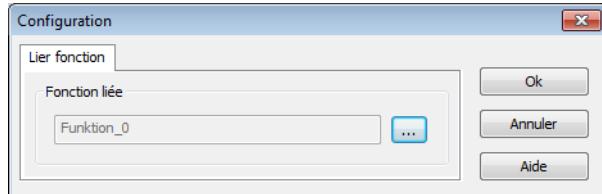


| Parameter           | Description                                                                                                                                      |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>         | Name of the property.                                                                                                                            |
| <b>Linkage</b>      | Linked function. Clicking in the cell opens the configuration dialog.                                                                            |
| <b>Type of link</b> | Selection of linking. Clicking in the cell opens the selection dialog.                                                                           |
| <b>WPF info</b>     | Shows the current value for properties in WPF content. For the user, it is directly visible what type of property it is (Boolean, string, etc.). |
| <b>Linked</b>       | Shows if a property is currently being used.<br>If not contained in the view, can be selected via <b>Context menu-&gt;Column selection</b> .     |

## LINK FUNCTIONS

To create a link:

1. Highlight the line with the property that is to be linked
2. Click in the **Link type cell**
3. Select from the drop down list function
4. Click in the **Link** cell
5. The dialog for configuring the function opens

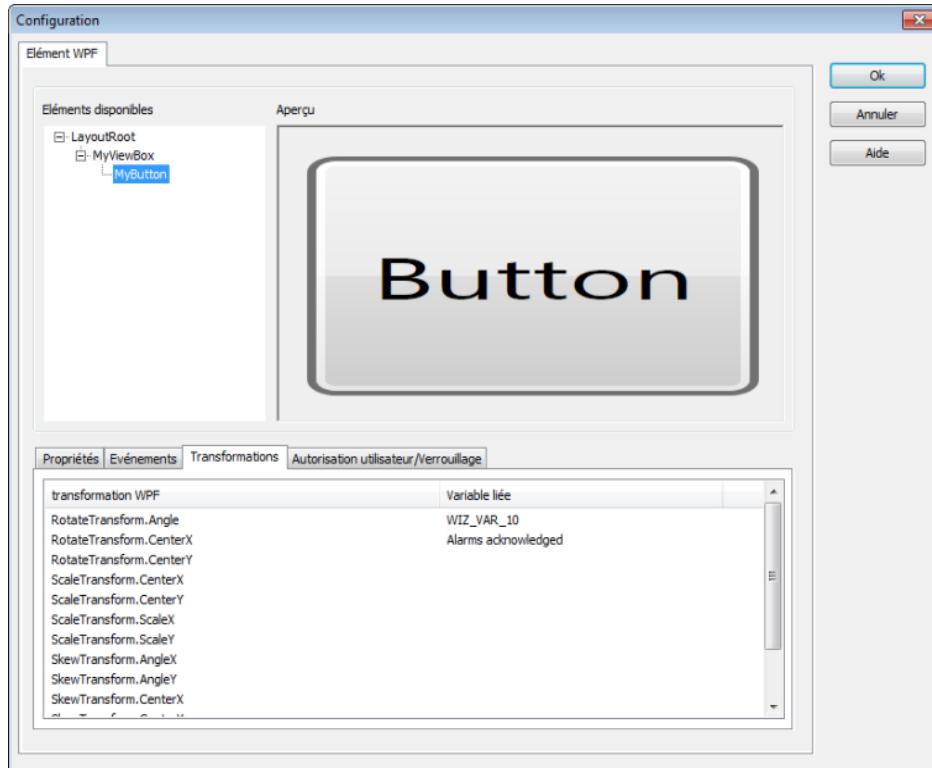


| Parameter              | Description                                                                                                 |
|------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Linked function</b> | Selection of the function to be linked. Clicking on the ... button opens the dialog for Function selection. |
| <b>OK</b>              | Accepts selection and closes dialog.                                                                        |
| <b>Cancel</b>          | Discards changes and closes dialog.                                                                         |
| Help                   | Opens online help.                                                                                          |

### 6.3.3 Transformation

The **WPF element** does not support rotation. If, for example, the **WPF element** is in a symbol and the symbol is rotated, the **WPF element** does not rotate with it. Therefore there is a different mechanism for **Transformation** with WPF to turn elements or to otherwise transform them. These transformations are configured in the **Transformation** tab.

**Attention:** If the content is outside of the **WPF element** area, this part of the contents is lost or it is not shown.



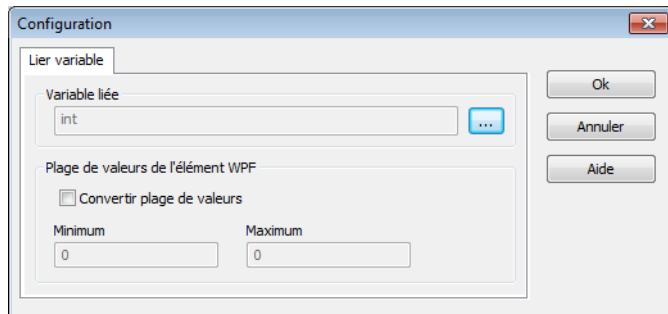
| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>         | Name of the property.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Linkage</b>      | <p>Selection of the linked variables.</p> <p>Transformations are displayed in XAML as transformation objects with their own properties. If an element supports a transformation, then the possible properties of the transformation object are displayed in list view. (more on this in: Integrate button as WPF XAML in zenon (à la page 118)</p> <p>For example, if the linked variable is set at the value of 10, then this value is written as a WPF target and the WPF element is rotated by 10°.</p> |
| <b>Type of link</b> | Selection of transformation link type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>WPF info</b>     | Shows the current value for properties in WPF content. For the user, it is directly visible what type of property it is (Boolean, string, etc.).                                                                                                                                                                                                                                                                                                                                                           |
| <b>Linked</b>       | <p>Shows if a property is currently being used.</p> <p>If not contained in the view, can be selected via <b>Context menu-&gt;Column selection</b>.</p>                                                                                                                                                                                                                                                                                                                                                     |

## LINK TRANSFORMATIONS

To link a transformation with a WPF property:

1. Highlight the line with the property that is to be linked
2. Click in the **Link type cell**
3. Select from the **Transformation** drop down list
4. Click in the **Link** cell
5. The dialog for configuring the variables opens

The configuration also makes it possible to convert from zenon into WPF units.



| Parameter                                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Linked variable</b>                    | Selection of the variable to be linked. A click on the ... button opens the selection dialog.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Range of values of the WPF element</b> | Data to convert variable values into WPF values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Convert range of values</b>            | <p><b>Active:</b> WPF unit conversion is switched on.</p> <p><b>Effect in the Runtime:</b> The current zenon value (incl. zenon unit) is converted to the WPF range using standardized minimum and maximum values.</p> <p><b>For example:</b> The value of a variable varies from 100 to 200. With the variables, the standardized range is set to 100 - 200. The aim is to display this change in value using a WPF rotary knob. For this:</p> <ul style="list-style-type: none"> <li>▶ for <b>Transformations</b>, the <b>RotateTransform.Angle</b> property is linked to the variables</li> <li>▶ <b>Adjust value</b> activated</li> <li>▶ a WPF value range of 0 to 360 is configured</li> </ul> <p>Now the rotary knob can be turned at a value of</p> |

| Parameter      | Description                            |
|----------------|----------------------------------------|
|                | 150, for example, by 180 degrees.      |
| <b>Minimum</b> | Defines the lowest WPF value.          |
| <b>Maximum</b> | Defines the highest WPF value.         |
| <b>OK</b>      | Accepts settings and ends the dialog.  |
| <b>Cancel</b>  | Discards settings and ends the dialog. |
| <b>Help</b>    | Opens online help.                     |

## 6.4 Validity of XAML Files

XAML files are valid subject to certain requirements:

- ▶ Correct name spaces
- ▶ No class references
- ▶ Scalability

### CORRECT NAME SPACE

The **WPF element** can only display WPF content, i.e.:

Only XAML files with the correct WPF namespace can be displayed by the **WPF element**. Files that use a Silverlight namespace cannot be loaded or displayed. However, in most cases it is suffice to change the Silverlight namespace to the WPF namespace.

WPF-Namespace:

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

### NO USE OF CLASS REFERENCES

Because the XAML files can be loaded dynamically, it is not possible to use XAML files that contain references to classes ("class" key in header). Functions that have been programmed in independently-created C#- files cannot be used.

In order to use WPF user controls with code behind, the process as described in the Creating a simple WPF user control with code behind funciton (à la page 24) must be carried out.

## SCALABILITY

If the content of a **WPF element** is adjusted to the size of the **WPF element**, then the controls of the **WPF element** are interlaced in a control that offers this functionality, such as a **view box** for example. In addition, it must be ensured that the **height** and **width** for this elements are configured as **automatic**.

## CHECKING AN XAML FILE TO SEE IF IT IS CORRECT

To check if an XAML file has the correct format:

- ▶ Open XAML file in Internet Explorer
  - ▶ If it can be opened without additional plug-ins (Java or similar), then it can be assumed with a high degree of certainty that this file can be loaded or displayed by zenon
  - ▶ if problems occur during loading, these are then shown in Internet Explorer and the lines in which problems arise can be clearly seen

The scaling can also be tested in this manner: If the file has been created correctly, the content will adjust to the size of the Internet Explorer window.

## ERROR MESSAGE

If an invalid file is used in zenon, then an error message is displayed in the output window when loading the file in the WPF element.

For example:

```
"error when loading
xaml-Datei:C:\ProgramData\COPA-DATA\SQL\781b1352-59d0-437e-a173-08563c3142e9\FILES\zenon\cu
stom\media\UserControl1.xaml
```

*The attribute "Class" cannot be found in XML namespace  
"http://schemas.microsoft.com/winfx/2006/xaml". Line 7 Position 2."*

## 6.5 Pre-built elements

Registered zenon partners can obtain examples of WPFs via **MyArea** (<https://www.copadata.com/en/my-area/overview/>) on the COPA-DATA website.

zenon is already shipped with several WPF elements. More are available for download in the web shop.

All WPF elements have properties which determine the graphical design of the respective element (Dependency Properties). Setting the values via an XAML file or linking the property via zenon can directly change the look in the Runtime. The tables in the description of the individual elements contain the respective Dependency Properties, depending on the control.

Available elements:

- ▶ Analog clock (à la page 64)
- ▶ Vertical bar graph (à la page 65)
- ▶ BACnet WPF Control
- ▶ Comtrade Viewer (à la page 66)
- ▶ Energy class diagram (à la page 78)
- ▶ Progress bar (à la page 66)
- ▶ Pareto diagram (à la page 79)
- ▶ Sankey Diagram (à la page 86)
- ▶ Round display (à la page 82)
- ▶ Temperature control (à la page 87)
- ▶ Universal slider (à la page 89)
- ▶ Waterfall chart (à la page 91)

## REPLACING ASSEMBLY WITH A NEWER VERSION

Per project only one *Assembly* for a WPF element can be used in the zenon Editor as well as in the Runtime. If two versions of an *Assembly* are available in a project, then the first loaded file is used. A user enquiry is made as to which version should be used. No further actions are needed for the maintenance of the versions used up until now. If a newer version is chosen, all corresponding CDWPF files in all symbols and images in all projects must be adapted.

**Note for Multi-Project Administration:** If an *Assembly* in a project is replaced by a new version, it must also be replaced in all other projects that are loaded in the Editor or in Runtime.

### 6.5.1 Analog clock - AnalogClockControl

| Property                      | Function                                  | Value                                                                                                             |
|-------------------------------|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>ElementStyle</b>           | Shape/type of element.                    | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ SmallNumbers</li> <li>▶ BigNumberS</li> <li>▶ No</li> </ul> |
| <b>ElementBackgroundBrush</b> | Color of element background.              | <i>Brush</i>                                                                                                      |
| <b>ElementGlasReflection</b>  | Activate the glass effect on the element. | <i>Visibility</i>                                                                                                 |

| Property          | Function                                                            | Value         |
|-------------------|---------------------------------------------------------------------|---------------|
| <b>Offset</b>     | Value in hours (h) which displays the time lag to the system clock. | <i>Int16</i>  |
| <b>OriginText</b> | Text which is displayed in the clock (e.g. location).               | <i>String</i> |

## 6.5.2 Bar graph vertical - VerticalBargraphControl

| Property                       | Function                                  | Value             |
|--------------------------------|-------------------------------------------|-------------------|
| <b>CurrentValue</b>            | Current value which should be displayed.  | <i>Double</i>     |
| <b>MinValue</b>                | Minimum value of the scale.               | <i>Double</i>     |
| <b>MaxValue</b>                | Maximum value of the scale.               | <i>Double</i>     |
| <b>MajorTicksCount</b>         | Number of main ticks on the scale.        | <i>Integer</i>    |
| <b>MinorTicksCount</b>         | Number of sub ticks on the scale.         | <i>Integer</i>    |
| <b>MajorTickColor</b>          | Color of main ticks on the scale.         | <i>Color</i>      |
| <b>MinorTickColor</b>          | Color of sub ticks on the scale.          | <i>Color</i>      |
| <b>ElementBorderBrush</b>      | Color of the element border.              | <i>Brush</i>      |
| <b>ElementBackgroundBrush</b>  | Color of element background.              | <i>Brush</i>      |
| <b>ElementGlasReflection</b>   | Activate the glass effect on the element. | <i>Visibility</i> |
| <b>ElementFontFamily</b>       | Element font.                             | <i>Font</i>       |
| <b>ScaleFontSize</b>           | Font size of the scale.                   | <i>Double</i>     |
| <b>ScaleFontColor</b>          | Font color of the scale.                  | <i>Color</i>      |
| <b>IndicatorBrush</b>          | Bar graph fill color.                     | <i>Brush</i>      |
| <b>BargraphSeparation</b>      | Number of bar graph division.             | <i>Integer</i>    |
| <b>BargraphSeparationColor</b> | Color of the scale division.              | <i>Color</i>      |

### 6.5.3 Progress bar - ProgressBarControl

| Property                         | Function                                                                         | Value          |
|----------------------------------|----------------------------------------------------------------------------------|----------------|
| <b>CurrentValue</b>              | Current value which should be displayed.                                         | <i>Double</i>  |
| <b>MinValue</b>                  | Minimum value of the value area.                                                 | <i>Double</i>  |
| <b>MaxValue</b>                  | Maximum value of the value area.                                                 | <i>Double</i>  |
| <b>ProgressbarDivisionCount</b>  | Number of divisions of the progress bar.                                         | <i>Integer</i> |
| <b>VisibilityText</b>            | Visibility of the value display.                                                 | <i>Boolean</i> |
| <b>FontSize</b>                  | Font size of the value display.                                                  | <i>Double</i>  |
| <b>TextColor</b>                 | Color of the value display.                                                      | <i>Color</i>   |
| <b>ProgressBarBoxedColor</b>     | Color of the border of the progress bar.                                         | <i>Color</i>   |
| <b>ProgressBarMarginDistance</b> | Distance of the progress bar box from the element edge (left, top, right, down). | <i>Double</i>  |
| <b>ProgressBarInactiveBrush</b>  | Indicator color not active.                                                      | <i>Brush</i>   |
| <b>ProgressBarActiveBrush</b>    | Indicator color active.                                                          | <i>Brush</i>   |
| <b>ProgressBarPadding</b>        | Distance of the progress bar from the progress bar box (left, top, right, down). | <i>Double</i>  |
| <b>ElementBorderBrush</b>        | Color of the element border.                                                     | <i>Brush</i>   |
| <b>ElementBackgroundBrush</b>    | Color of element background.                                                     | <i>Brush</i>   |

### 6.5.4 COMTRADE-Viewer

The **COMTRADE-Viewer** WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).

#### Informations de licence

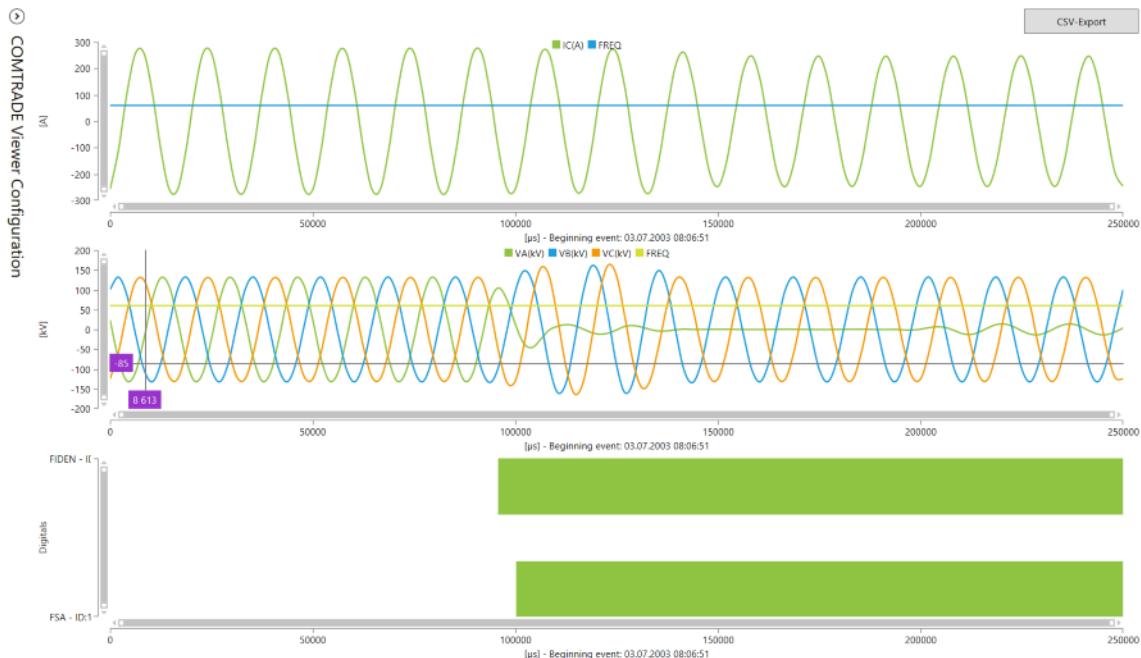
The **COMTRADE-Viewer** can only be configured in the zenon Editor with a valid Energy Edition license. If there is no valid license, the WPF is displayed as grayed out in the Editor. A valid Energy Edition license is also required for display in the Runtime.

The control supports the graphical analysis of digital error and event data logging of a COMTRADE file.

## Information

The control supports IEEE C37.111 (IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems) standards-compliant files. ASCII or binary files in accordance with the 1999 or 2013 edition can be visualized.

Older files or files without a year identification are not supported. A warning dialog is called up when an invalid/unsupported file is selected.



Possibilities of the **COMTRADE-Viewer** WPF control in zenon Runtime:

- ▶ Selection of a file in the COMTRADE file format
- ▶ Visualization of the selected COMTRADE file:
  - Note:** The display colors can be configured in the zenon Editor.
  - ▶ Current (sinus wave display)
  - ▶ Voltage (sinus wave display)
  - ▶ Digital signals (binary bar chart display)
  - ▶ Display of values at a selected cursor position.
  - ▶ If an element that represents neither current or voltage is selected, (such as frequency), this is visualized in both analog areas again (current and voltage).
- ▶ Navigation:
  - ▶ Zoom in and zoom out using the mouse wheel, scroll bar and Multi-Touch gestures

- ▶ Enlargement of the area  
Selection of the area by clicking the mouse
- ▶ Move the display area using the right mouse button, scroll bar or Multi-Touch gestures.
- ▶ Exports selected objects as an CSV file.

### Hint

To be able to transport COMTRADE files to the zenon Runtime computer, you can also use the file transfer of the **IEC 61850 driver** or the **FTP function block** of zenon Logic.

You can find further information about this in the driver documentation of the IEC 61850 driver or in the zenon Logic documentation.

#### 6.5.4.1 View in Runtime

The **COMTRADE** WPF element offers two views in the Runtime:

- ▶ Configuration view
  - ▶ Selection of a COMTRADE configuration file
  - ▶ Selection of the elements to be displayed
- ▶ Graph view
  - ▶ Zoom in and zoom out
  - ▶ Display of values at a selected cursor position.
  - ▶ Export of the selected elements as an CSV file

### Information

The switch between the views is integrated in the WPF element. Additional project configuration of a screen switching function is not necessary.

#### 6.5.4.2 Installation

Requirements:

- ▶ You must be a registered COPA-DATA Partner.
- ▶ You need to have access to the COPA-DATA Partner Community already granted.

## TO DOWNLOAD A WPF ELEMENT:

1. Open the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).
2. Log in in the area called **MY AREA**.
3. Enter "Partner" as a search term by the microscope.  
    **COPA-DATA Partner Community** appears with a link in the results.
4. Click on the link.
5. In the **Partner Login Area**, click on **MORE**.  
    The **Overview** tab is opened.
6. Switch to the **Documents & Downloads** tab.
7. Under **SELECT CATEGORY**, activate the **WPF element** checkbox.
8. Click on **Search**.  
    The labeling is hidden.  
    An orange frame surrounds the search field.
9. Click in the frame and enter **COMTRADE**.
10. The **COMTRADE Viewer** is shown.
11. Under **Actions**, click on the orange symbol to download the **COMTRADE Viewer**.
12. Save the folder locally.
13. Unzip the file.
14. Open the **COMTRADE** folder that contains the **Comtrade.cdwpf** file.

### 6.5.4.3 Installation

To create a WPF element

1. Open the zenon Editor
2. In the project manager, go to **Files** and **Graphics**.
3. Go to **Import file....**
4. Select the file **Comtrade.cdwpf** for importing.
5. Confirm the selection by clicking on **Open**.
6. Open the screen in which the WPF element is to be inserted.
7. In the vertical menu bar, select the WPF element (**WPF**).
8. Drag the WPF element in the screen with the left mouse button.  
    The **File selection** window is opened.

9. Select the file **Comtrade.cdwpf** and confirm the input by clicking on **OK**.

The WPF element is added.

#### 6.5.4.4 Propriétés configurables du contrôle- affichage des couleurs

##### CONFIGURATION DANS EDITOR

L'élément nommé **COMTRADE.CDWPF** peut être configuré et placé dans tous les types de synoptiques zenon.

La configuration de la **Largeur [pixels]** et la **Hauteur [pixels]** de l'élément dans le projet dépend des proportions. Ceci évite toute distorsion de l'affichage **COMTRADE-Viewer** dans le Runtime.

**Remarque :** Lors de la configuration du projet, assurez-vous de rendre cet élément suffisamment grand pour garantir sa lisibilité.

##### MODIFICATIONS GRAPHIQUES

L'apparence graphique est configurée dans les propriétés de l'élément WPF.

D'autres informations sont disponibles au chapitre Configuration des liaisons (à la page 49) dans ce manuel.

Valeurs possibles des couleurs :

- ▶ Des valeurs de couleurs hexadécimales aux formats suivants peuvent être utilisées :  
#RRGGBB et #AARRGGBB  
Les valeurs de transparence sont autorisées.
- ▶ **Exemples de valeurs de couleurs :**  
#000000 = noir  
#FFFFFF = blanc  
#FF0000 = rouge
- ▶ Valeurs de couleurs par nom  
Référence : <https://msdn.microsoft.com/en-us/library/system.drawing.color.aspx>  
(<https://msdn.microsoft.com/en-us/library/system.drawing.color.aspx>)

##### Hint

La couleur de départ des propriétés de l'élément WPF **COMTRADE-Viewer** est "z". Utilisez le filtrage par nom pour obtenir un affichage clair lors de la configuration des liaisons.

## PAGE DE CONFIGURATION

Couleur du texte et de l'arrière-plan de la page de configuration.

### Analog Channels

| Paramètre                                | Description                                        | Valeur        |
|------------------------------------------|----------------------------------------------------|---------------|
| <b>zConfiguratinPageTextColor</b>        | Couleur du texte de la page de configuration       | <i>String</i> |
| <b>zConfigurationPageBackgroundColor</b> | Couleur d'arrière-plan de la page de configuration | <i>String</i> |

## BOUTONS

Couleur du texte et de l'arrière-plan du bouton.

### Open...

| Paramètre                     | Description                      | Valeur        |
|-------------------------------|----------------------------------|---------------|
| <b>zButtonTextColor</b>       | Couleur du texte du bouton       | <i>String</i> |
| <b>zButtonBackgroundColor</b> | Couleur d'arrière-plan du bouton | <i>String</i> |

## DIAGRAMME

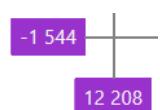
Couleur du texte des intitulés des axes ou couleur de la légende et de l'arrière-plan.

50000                    60000                    70000

| Paramètre                    | Description                                       | Valeur        |
|------------------------------|---------------------------------------------------|---------------|
| <b>zChartTextColor</b>       | Couleur du texte des intitulés des axes.          | <i>String</i> |
| <b>zChartBackgroundColor</b> | Couleur de l'arrière-plan des intitulés des axes. | <i>String</i> |

## INTITULÉ

Couleur du texte et de l'arrière-plan de l'affichage d'une position de pointeur sélectionnée.



| Paramètre                         | Description                                       | Valeur        |
|-----------------------------------|---------------------------------------------------|---------------|
| <b>zChartLabelTextColor</b>       | Couleur du texte de l'affichage de la valeur      | <i>String</i> |
| <b>zChartLabelBackgroundColor</b> | Couleur d'arrière-plan de l'affichage des valeurs | <i>String</i> |

## DIAGRAMME

Palette de couleurs de l'affichage graphique et des légendes associées.

| Paramètre            | Description                                                                                                                                                                                                                                                         | Valeur        |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>zChartPalette</b> | Palette de couleurs des graphiques et des légendes.<br>Référencement avec le nom de la palette de couleurs (voir la vue d'ensemble).<br>Par défaut : si aucune palette de couleurs n'est configurée, la palette de couleurs du système d'exploitation est utilisée. | <i>String</i> |

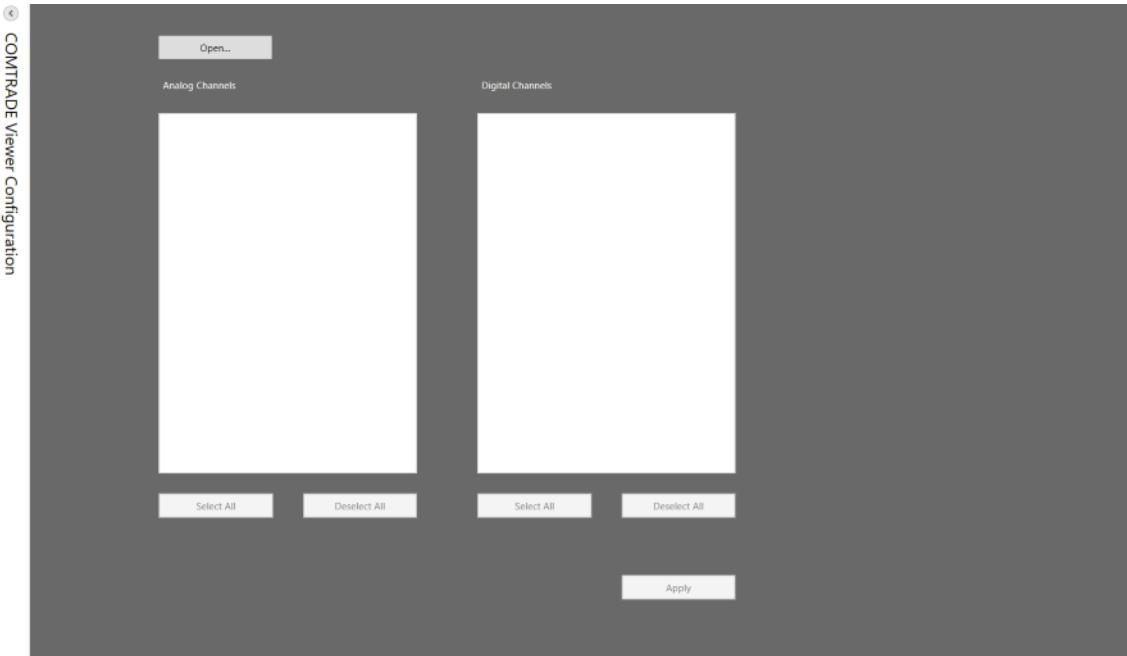
## PALETTES DE COULEURS POSSIBLES - APERÇU

|           |                                                                                     |
|-----------|-------------------------------------------------------------------------------------|
| Arctic    |  |
| Autumn    |  |
| Cold      |  |
| Flower    |  |
| Forest    |  |
| Grayscale |  |
| Ground    |  |
| Lilac     |  |
| Natural   |  |
| Pastel    |  |
| Rainbow   |  |
| Spring    |  |
| Summer    |  |
| Warm      |  |
| Windows8  |  |

### 6.5.4.5 Runtime view - configuration page

If a screen with a configured **COMTRADE-Viewer** WPF element is called up, the display of the respective configuration page is empty.

**Note:** This also applies if, in zenon Runtime, there is a switch from one screen to another screen with the screen switching function.



#### COMTRADE VIEWER CONFIGURATION

The **COMTRADE Viewer Configuration** switching, arranged vertically on the side, switches the display of the configuration to graphic view and vice versa.

#### SELECT FILE

The **Open...** button opens the file selection dialog to select a file.

There is a pre-selection for display in the file selection:

- ▶ In doing so, file pairs of \*.cfg- and \*.dat files are detected.
- Note:** Optional \*.hdr or \*.inf files are not taken into account.
- ▶ Only the corresponding \*.dat files are displayed.
- ▶ All attendant files (\*.dat, \*.cfg) are loaded by clicking on the desired file and the **OK** button.
- ▶ One file can be loaded.
- ▶ After loading the file, the contents of the file are shown in the **Analog Channels** and **Digital Channels** columns.

The labels and units of the elements originate from the **COMTRADE** configuration and cannot be changed.

## FURTHER INFORMATION ON THE EDITING OF \*.CFG- AND \*.DAT FILES

The information from the \*.cfg file allows the evaluation of the \*.dat file. It contains the data from various analog and digital series of measurements of currents and voltages. The data is broken down into individual data sets and shown in hex format.

### \*.cfg files

- ▶ The last entry of a file of this data type is a time multiplier. This entry is multiplied by the time stamp of one of each entry from the \*.dat file when a disturbance is read in. If there is no time multiplier, a factor with the value of 1 is assumed internally. The \*.cfg file is not changed in the process.
- ▶ Certain standards apply for the entries of the digital measured values. Example of a standard-compliant entry of a digital measured value: 1,LOPHC,,,0. However, if there is no zero at the end of the entry, the **COMTRADE-Viewer** adds this internally. The \*.cfg file is not changed in the process.

### \*.dat files

- ▶ The **COMTRADE-Viewer** is in a position to read in files of this data type that start with the index 0 or >1. In doing so, a check is constantly carried out to see whether these data sets are numbered continually in discrete steps from 1. If there are data sets that are not correctly numbered, the file cannot be read in.

## ANALOG CHANNELS

| Parameter                      | Description                                                                                                                                                                                                                       |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Liste der verfügbaren Kanäle] | Selection of the elements to be visualized.<br><br>Multiple selection by clicking on the desired entry in the list. Selected elements are shown with a colored background. Another mouse click undoes the selection of the entry. |
| Select All                     | Selects all elements from the list.                                                                                                                                                                                               |
| Deselect All                   | Deactivates the existing selection of elements.                                                                                                                                                                                   |

## DIGITAL CHANNELS

| Parameter                      | Description                                |
|--------------------------------|--------------------------------------------|
| [Liste der verfügbaren Kanäle] | Selection of the elements to be visualized |

| Parameter    | Description                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | Multiple selection by clicking on the desired entry in the list. Selected elements are shown with a colored background. Another mouse click undoes the selection of the entry. |
| Select All   | Selects all elements from the list.                                                                                                                                            |
| Deselect All | Deactivates the existing selection of elements.                                                                                                                                |

## SHOW SELECTION

To show your selection in the graphic view, click on the **Apply** button.

**Note:** Clicking on the vertically-arranged **COMTRADE Viewer Configuration** switching only changes the view. An amended selection of the channels is not taken into account in the process.

### 6.5.4.6 Runtime view - visualization of COMTRADE data

The selected channels are visualized in the graph view of the **COMTRADE-Viewer** WPF element. The coloring can be configured in the zenon Editor.

Files that are compliant with the **IEEE C37.111-2013** (or **IEC 60255-24:2013**) standard can be read in.

X-values are necessary for the display, which represent the time stamps of the respective recorded data points.

The X-values can be saved in the **\*.dat** file directly.

If this is not the case, there is a calculation of the time stamp using the following entries of the **\*.cfg** file:

- ▶ Start time of recording the disturbance (date and time of the first data point)
- ▶ Number of recorded data points (**Samplerate**)
- ▶ Number of data points for each query (**Rate**)

## EXPORT OF THE SELECTED DATA

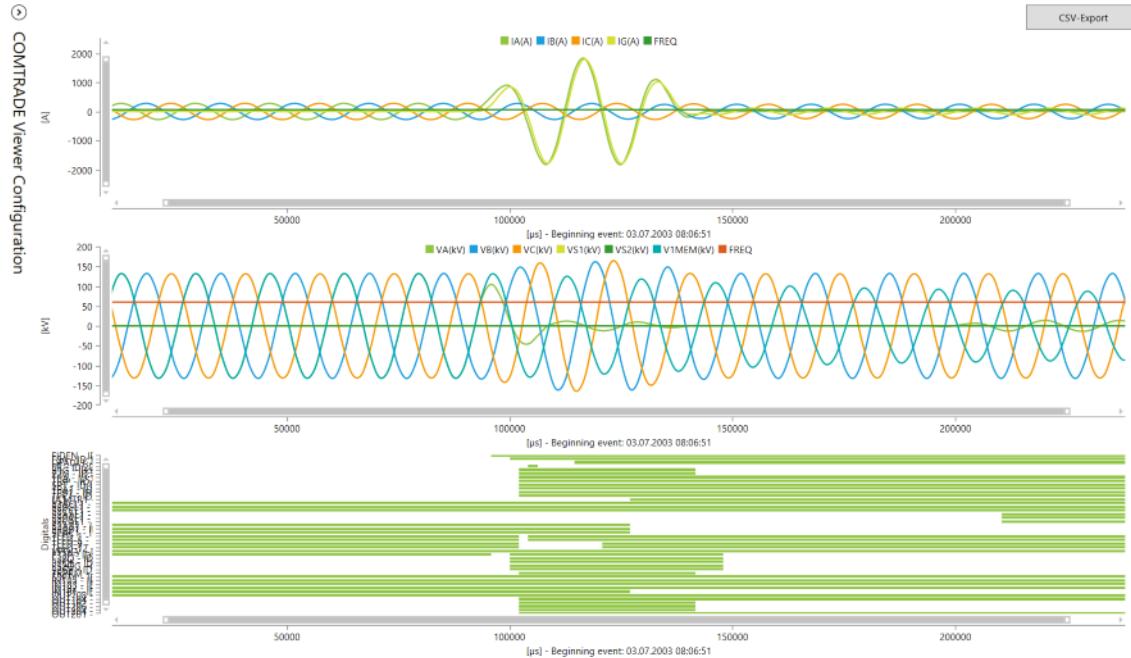
The selected analog and digital channels can be exported to a CSV file with the **CSV-Export** button.

## GRAPH VIEW

The graph view of the **COMTRADE-Viewer** is divided into three sections:

- ▶ Current amperage
- Upper area

- ▶ Voltage  
Mid area
- ▶ Digital channels  
Lower area



## AXIS LABELING

- ▶ Horizontal axis  
The horizontal axis represents the complete time period as illustrated in the COMTRADE file (\*.dat).  
The scaling of this time axis depends on the enlargement level. The higher the enlargement selected, the more detailed the time display.
- ▶ Vertical axis  
The vertical axis represents the values.
  - ▶ The scaling of the value axis depends on the enlargement level. The greater the enlargement selected, the more detailed the display of values.
  - ▶ The labeling of the analog channels is shown vertically next to the values and corresponds to the measuring unit as defined in the COMTRADE file (\*.cfg).
  - ▶ The digital channels are displayed in the sequence as defined in the COMTRADE file (\*.cfg). The *Channel identifier* of the COMTRADE file serves as an identifier.

## KEY

■ IA(A) ■ IB(A) ■ IC(A) ■ IG(A)

The color key of the graphs is shown at the head of the graph.

- ▶ The labeling of the digital channels corresponds to the channel description as defined in the COMTRADE file (\*.cfg).
- ▶ The colors for each channel are assigned automatically with the configured color palette.
- ▶ The time is displayed in a footer under the graph. The start time is displayed as a text.

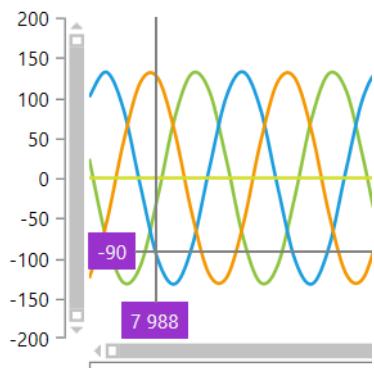
## NAVIGATION AND ZOOM

Navigation (scroll and zoom) is always applied to all three areas of the graphic display.

- ▶ You can move the display within the horizontal time line with the scroll bar.
- ▶ Zoom in and zoom out
  - ▶ You can zoom at the current position of the mouse pointer in the graph view or reduce the enlargement.
  - ▶ The selected area is displayed by selecting a display area with the mouse button held down.
- ▶ Double clicking on the scroll bar resets the enlargement.

**Note:** The display of the values is always amended to the selected area. As a result, this can lead to a flattening of the curve in the enlarged graph view.

## ANALYSIS



The precise values at the position of the mouse pointer are visualized with a display in value blocks. A crosshair offers additional visual support with the exact determination of the reading position.

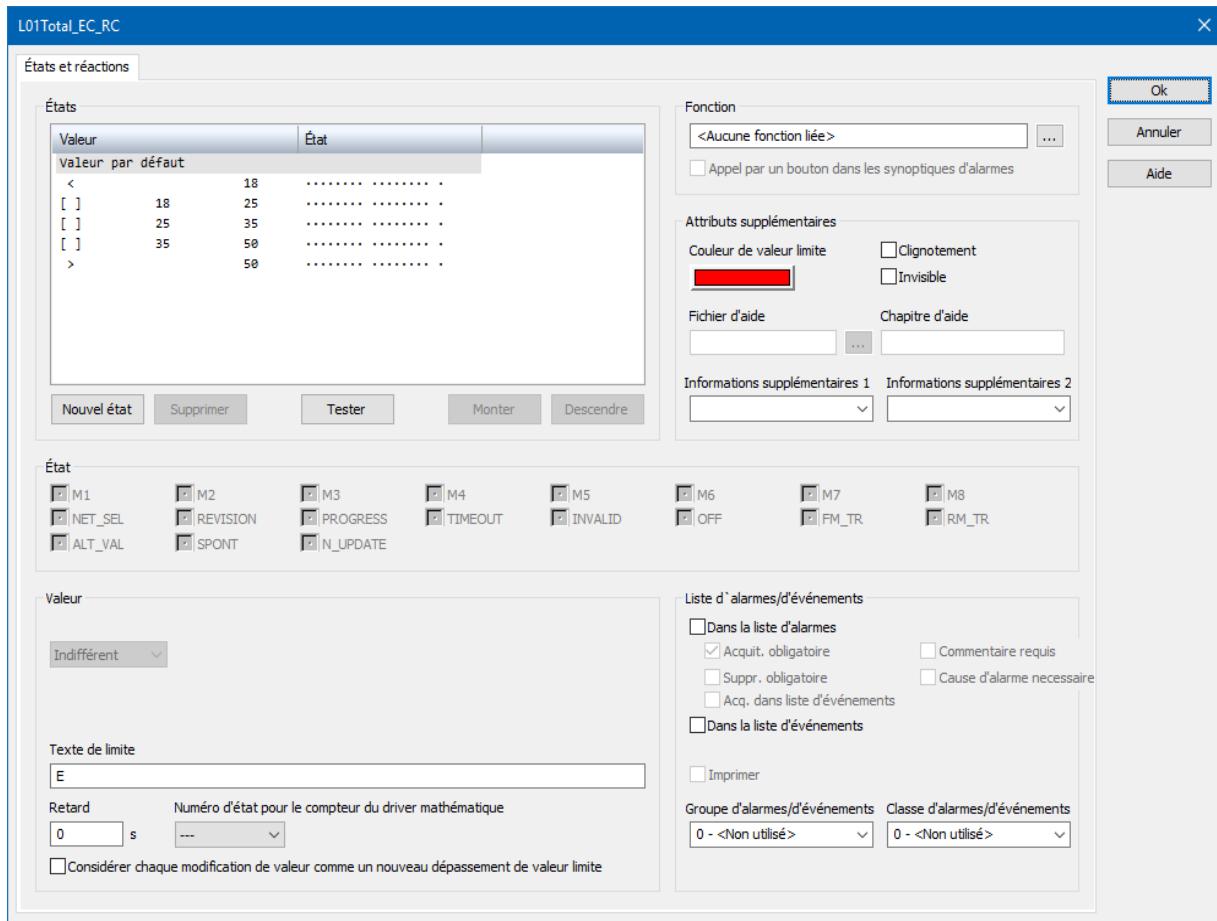
## 6.5.5 Energy class diagram

The energy class diagram, WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).



A reaction matrix must be used to model an energy class diagram. This reaction matrix must be linked to the variable whose value is envisaged for display and distribution in energy classes. The name of the variable must be transferred to the "**zVariableName**" property.

### REACTION MATRIX FOR ENERGY CLASS DIAGRAM



L01Total\_EC\_RC

États et réactions

États

| Valeur | État |
|--------|------|
| <      | 18   |
| [ ]    | 25   |
| [ ]    | 35   |
| [ ]    | 50   |
| >      | 50   |

Nouvel état Supprimer Tester Monter Descendre

Fonction

<Aucune fonction liée> ...

Appel par un bouton dans les synoptiques d'alarmes

Attributs supplémentaires

Couleur de valeur limite   Clignotement  Invisible

Fichier d'aide Chapitre d'aide

Informations supplémentaires 1 Informations supplémentaires 2

État

|         |          |          |         |         |     |       |       |
|---------|----------|----------|---------|---------|-----|-------|-------|
| M1      | M2       | M3       | M4      | M5      | M6  | M7    | M8    |
| NET_SEL | REVISION | PROGRESS | TIMEOUT | INVALID | OFF | FM_TR | RM_TR |
| ALT_VAL | SPONT    | N_UPDATE |         |         |     |       |       |

Valeur

Indifférent

Texte de limite

E

Retard Numéro d'état pour le compteur du driver mathématique

0 s ---

Considérer chaque modification de valeur comme un nouveau dépassement de valeur limite

Liste d'alarmes/d'événements

Dans la liste d'alarmes  Acquit. obligatoire  Suppr. obligatoire  Acq. dans liste d'événements  Commentaire requis  Cause d'alarme nécessaire

Dans la liste d'événements

Imprimer

Groupe d'alarmes/d'événements Classe d'alarmes/d'événements

0 - <Non utilisé> 0 - <Non utilisé>

The linked reaction matrix must correspond to the following schematic:

- ▶ The first status must be an area, or a "less than" definition
- ▶ Then as many different areas as desired can be defined.

- The last status must be an area or a "greater than" definition.

The following is applicable for project configuration:

1. If the first status is an area and the value of the variable comes under this area, the first status in the diagram is shown nevertheless. The same is applicable for the last status the other way round.
2. The colors that the WPF diagram uses for the classes are the limit value colors that were defined in the reaction matrix.
3. The letters for the classes are set in alphabetical order starting with "A".

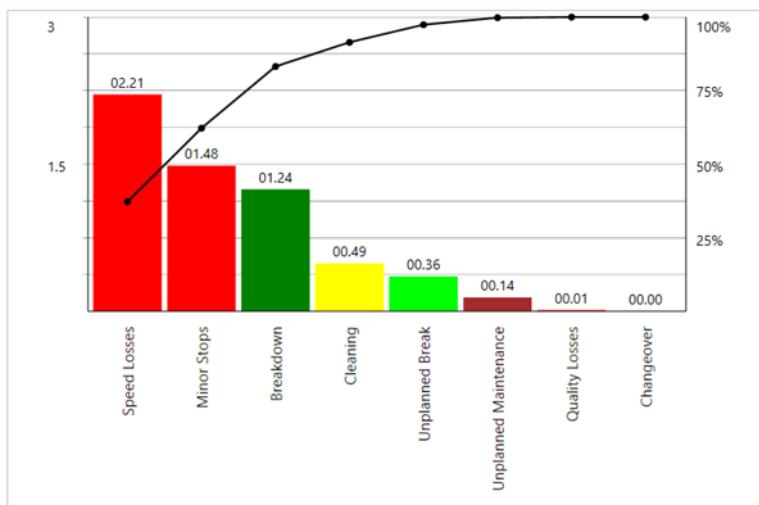
| Property                          | Description                                                                  | Value   |
|-----------------------------------|------------------------------------------------------------------------------|---------|
| <b>zenonFontID</b>                | ID for a font from the first font list (font size is not taken into account) | Integer |
| <b>zenonNumberOfDecimalPlaces</b> | Number of displayed decimal points                                           | Integer |
| <b>zenonVariableName</b>          | Name of the variable to be displayed.                                        | String  |

**Note:** Additional VSTA programming is necessary for the display of the energy class diagram in the zenon web client. You can find details on this in the display of WPF elements in the zenon web client (à la page 110).

## 6.5.6 Diagramme de Pareto

Le diagramme de Pareto, WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).

Un exemple de diagramme de Pareto dans le Runtime est illustré ci-dessous :



Les paramètres suivants peuvent être définis dans la fenêtre de configuration WPF sous **COPADATA-ELEMENT** :

| Propriété                       | Fonction                                                                | Valeur                         |
|---------------------------------|-------------------------------------------------------------------------|--------------------------------|
| <b>zenonBarColor1</b>           | Couleur de la première barre                                            | COLOR (STRING)                 |
| <b>zenonBarColor2</b>           | Couleur de la deuxième barre                                            | COLOR (STRING)                 |
| <b>zenonBarColor3</b>           | Couleur de la troisième barre                                           | COLOR (STRING)                 |
| <b>zenonBarColor4</b>           | Couleur de la quatrième barre                                           | COLOR (STRING)                 |
| <b>zenonBarColor5</b>           | Couleur de la cinquième barre                                           | COLOR (STRING)                 |
| <b>zenonBarColor6</b>           | Couleur de la sixième barre                                             | COLOR (STRING)                 |
| <b>zenonBarColor7</b>           | Couleur de la septième barre                                            | COLOR (STRING)                 |
| <b>zenonBarColor8</b>           | Couleur de la huitième barre                                            | COLOR (STRING)                 |
| <b>zenonBarColor9</b>           | Couleur de la neuvième barre                                            | COLOR (STRING)                 |
| <b>zenonBarColor10</b>          | Couleur de la dixième barre                                             | COLOR (STRING)                 |
| <b>zenonColorPercentageLine</b> | Couleur de la ligne de pourcentage (somme des fréquences relatives).    | COLOR (STRING)                 |
| <b>zenonInitialisationTime</b>  | Indique le délai d'attente d'initialisation des contrôles.              | INTEGER<br>Par défaut : 150 ms |
| <b>zenonLineVisibility</b>      | Visibilité de la ligne de pourcentage (somme des fréquences relatives). | BOOLEAN                        |
| <b>zenonVariable1_Label</b>     | Intitulé de la première barre                                           | STRING                         |
| <b>zenonVariable1_Value</b>     | Valeur de la première barre                                             | DOUBLE                         |
| <b>zenonVariable2_Label</b>     | Intitulé de la deuxième barre                                           | STRING                         |
| <b>zenonVariable2_Value</b>     | Valeur de la deuxième barre                                             | DOUBLE                         |
| <b>zenonVariable3_Label</b>     | Intitulé de la troisième barre                                          | STRING                         |
| <b>zenonVariable3_Value</b>     | Valeur de la troisième barre                                            | DOUBLE                         |
| <b>zenonVariable4_Label</b>     | Intitulé de la quatrième barre                                          | STRING                         |
| <b>zenonVariable4_Value</b>     | Valeur de la quatrième barre                                            | DOUBLE                         |
| <b>zenonVariable5_Label</b>     | Intitulé de la cinquième barre                                          | STRING                         |
| <b>zenonVariable5_Value</b>     | Valeur de la cinquième barre                                            | DOUBLE                         |

| Propriété                    | Fonction                      | Valeur        |
|------------------------------|-------------------------------|---------------|
| <b>zenonVariable6_Label</b>  | Intitulé de la sixième barre  | <i>STRING</i> |
| <b>zenonVariable6_Value</b>  | Valeur de la sixième barre    | <i>DOUBLE</i> |
| <b>zenonVariable7_Label</b>  | Intitulé de la septième barre | <i>STRING</i> |
| <b>zenonVariable7_Value</b>  | Valeur de la septième barre   | <i>DOUBLE</i> |
| <b>zenonVariable8_Label</b>  | Intitulé de la huitième barre | <i>STRING</i> |
| <b>zenonVariable8_Value</b>  | Valeur de la huitième barre   | <i>DOUBLE</i> |
| <b>zenonVariable9_Label</b>  | Intitulé de la neuvième barre | <i>STRING</i> |
| <b>zenonVariable9_Value</b>  | Valeur de la neuvième barre   | <i>DOUBLE</i> |
| <b>zenonVariable10_Label</b> | Intitulé de la dixième barre  | <i>STRING</i> |
| <b>zenonVariable10_Value</b> | Valeur de la dixième barre    | <i>DOUBLE</i> |

Les événements suivants peuvent être utilisés et liés aux fonctions de zenon :

| Événement             | Fonction                                                               | Valeur   |
|-----------------------|------------------------------------------------------------------------|----------|
| <b>zenonBar1Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la première barre.  | Fonction |
| <b>zenonBar2Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la deuxième barre.  | Fonction |
| <b>zenonBar3Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la troisième barre. | Fonction |
| <b>zenonBar4Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la quatrième barre. | Fonction |
| <b>zenonBar5Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la cinquième barre. | Fonction |
| <b>zenonBar6Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la sixième barre.   | Fonction |
| <b>zenonBar7Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la septième barre.  | Fonction |
| <b>zenonBar8Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la huitième barre.  | Fonction |
| <b>zenonBar9Click</b> | Fonction exécutée lorsque l'utilisateur                                | Fonction |

| Événement              | Fonction                                                             | Valeur   |
|------------------------|----------------------------------------------------------------------|----------|
|                        | clique sur la neuvième barre.                                        |          |
| <b>zenonBar10Click</b> | Fonction exécutée lorsque l'utilisateur clique sur la dixième barre. | Fonction |

### 6.5.7 Circular gauge control

| Property                      | Function                                         | Value                                                                                                                                     |
|-------------------------------|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CurrentValue</b>           | Current value which should be displayed.         | <i>Double</i>                                                                                                                             |
| <b>IsReversed</b>             | Scale orientation - clockwise or anti-clockwise. | <i>Boolean</i>                                                                                                                            |
| <b>ElementFontFamily</b>      | Element font.                                    | <i>Font</i>                                                                                                                               |
| <b>MinValue</b>               | Minimum value of the scale.                      | <i>Double</i>                                                                                                                             |
| <b>MaxValue</b>               | Maximum value of the scale.                      | <i>Double</i>                                                                                                                             |
| <b>ScaleRadius</b>            | Radius of the scale.                             | <i>Double</i>                                                                                                                             |
| <b>ScaleStartAngle</b>        | Angle at which the scale starts.                 | <i>Double</i>                                                                                                                             |
| <b>ScaleLabelRotationMode</b> | Alignment of the scale caption.                  | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ None</li> <li>▶ Automatic</li> <li>▶ Surround In</li> <li>▶ Surround Out</li> </ul> |
| <b>ScaleSweepAngle</b>        | Angle area which defines the size of the scale.  | <i>Double</i>                                                                                                                             |
| <b>ScaleLabelFontSize</b>     | Font size of the scale caption.                  | <i>Double</i>                                                                                                                             |
| <b>ScaleLabelColor</b>        | Font color of the scale caption.                 | <i>Color</i>                                                                                                                              |
| <b>ScaleLabelRadius</b>       | Radius on which the scale caption is orientated. | <i>Double</i>                                                                                                                             |
| <b>ScaleValuePrecision</b>    | Accuracy of the scale caption.                   | <i>Integer</i>                                                                                                                            |
| <b>PointerStyle</b>           | Shape of the pointer displaying the value.       | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ Arrow</li> </ul>                                                                    |

| Property                  | Function                               | Value                                                                                                                                                    |
|---------------------------|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                                        | <ul style="list-style-type: none"> <li>▶ <i>Rectangle</i></li> <li>▶ <i>TriangleCap</i></li> <li>▶ <i>Pentagon</i></li> <li>▶ <i>Triangle</i></li> </ul> |
| <b>MajorTickColor</b>     | Color of main ticks on the scale.      | <i>Color</i>                                                                                                                                             |
| <b>MinorTickColor</b>     | Color of sub ticks on the scale.       | <i>Color</i>                                                                                                                                             |
| <b>MajorTickSize</b>      | Size of main ticks on the scale.       | <i>Size</i>                                                                                                                                              |
| <b>MinorTickSize</b>      | Size of sub ticks on the scale.        | <i>Size</i>                                                                                                                                              |
| <b>MajorTicksCount</b>    | Number of main ticks on the scale.     | <i>Integer</i>                                                                                                                                           |
| <b>MajorTicksShape</b>    | Shape/type of main ticks on the scale. | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>Rectangle</i></li> <li>▶ <i>Trapezoid</i></li> <li>▶ <i>Triangle</i></li> </ul>                 |
| <b>MinorTicksShape</b>    | Shape/type of sub ticks on the scale.  | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>Rectangle</i></li> <li>▶ <i>Trapezoid</i></li> <li>▶ <i>Triangle</i></li> </ul>                 |
| <b>MinorTicksCount</b>    | Number of sub ticks on the scale.      | <i>Integer</i>                                                                                                                                           |
| <b>PointerSize</b>        | Size of the pointer.                   | <i>Size</i>                                                                                                                                              |
| <b>PointerCapRadius</b>   | Size of the pointer fastening point.   | <i>Double</i>                                                                                                                                            |
| <b>PointerBorderBrush</b> | Color of pointer border.               | <i>Brush</i>                                                                                                                                             |
| <b>PointerCapStyle</b>    | Shape/type of pointer fastening point. | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>BackCap</i></li> <li>▶ <i>FrontCap</i></li> </ul>                                               |

| Property                     | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Value         |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | ► Screw       |
| <b>PointerCapBorderBrush</b> | Color of pointer fastening point.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <i>Brush</i>  |
| <b>PointerBrush</b>          | Color of pointer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <i>Brush</i>  |
| <b>GaugeBorderBrush</b>      | Color of the element border.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>Brush</i>  |
| <b>GaugeBackgroundBrush</b>  | Color of element background.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>Brush</i>  |
| <b>PointerCapColorBrush</b>  | Color of pointer fastening point.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <i>Brush</i>  |
| <b>GaugeMiddlePlate</b>      | Radius of the element background middle plate.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>Double</i> |
| <b>PointerOffset</b>         | Offset of the pointer (displacement).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | <i>Double</i> |
| <b>RangeRadius</b>           | Radius of the total range display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Double</i> |
| <b>RangeThickness</b>        | Thickness of the total range display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | <i>Double</i> |
| <b>RangeStartValue</b>       | Start value of the total range display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <i>Double</i> |
| <b>RangexEndValue</b>        | <p>End value of an area. x stands for the range. The end value is at the same time the start value of the next respective range:</p> <ul style="list-style-type: none"> <li>► <b>Range1EndValue</b>: End value of the 1st area and start value of the 2nd range.</li> <li>► <b>Range2EndValue</b>: End value of the 2nd area and start value of the 3rd range.</li> <li>► <b>Range3EndValue</b>: End value of the 3rd area and start value of the 4th range.</li> <li>► <b>Range4EndValue</b>: End value of the 4th area and start value of the 5th range.</li> <li>► <b>Range5EndValue</b>: End value of the 5th area and start value of the 6th range.</li> <li>► <b>Range6EndValue</b>: End value of the 6th range.</li> </ul> | <i>Double</i> |
| <b>RangexColorBrush</b>      | <p>Color of the respective area. x stands for the range. Available are:</p> <ul style="list-style-type: none"> <li>► <b>Range1ColorBrush</b>: Color of the first range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Brush</i>  |

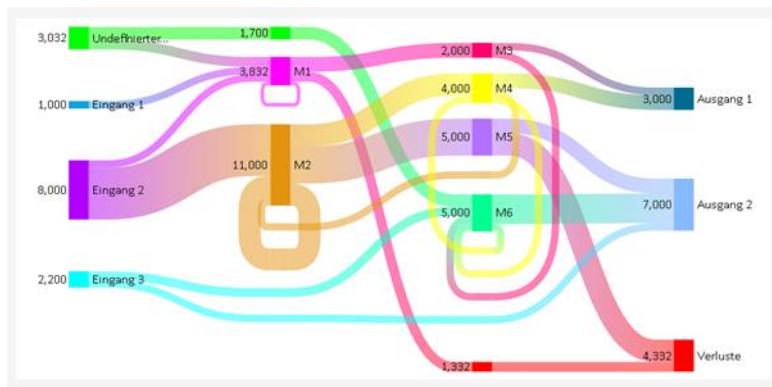
| Property                     | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Value                                                                                                                                  |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
|                              | <ul style="list-style-type: none"> <li>▶ <b>Range2ColorBrush</b>: Color of the second range.</li> <li>▶ <b>Range3ColorBrush</b>: Color of the third range.</li> <li>▶ <b>Range4ColorBrush</b>: Color of the fourth range.</li> <li>▶ <b>Range5ColorBrush</b>: Color of element fifth range.</li> <li>▶ <b>Range6ColorBrush</b>: Color of element sixth range.</li> </ul> <p><b>Note:</b> Cannot be set in zenon. You can find information on the use of CDWPF to change color values in the chapters <b>Referenced assemblies</b> (à la page 8), <b>CDWPF files</b> (à la page 48) and <b>Configurable control properties - color display</b> (à la page 70).</p> |                                                                                                                                        |
| <b>ScaleOuterBorderBrush</b> | Color of the scale border.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>Brush</i>                                                                                                                           |
| <b>ScaleBackgroundBrush</b>  | Color of scale background.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>Brush</i>                                                                                                                           |
| <b>ValueTextFrameStyle</b>   | Shape/type of value display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>LargeFrame</i></li> <li>▶ <i>SmallFrame</i></li> <li>▶ <i>None</i></li> </ul> |
| <b>ValueTextContent</b>      | Content of the value display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>Text</i></li> <li>▶ <i>TextValue</i></li> <li>▶ <i>Value</i></li> </ul>       |
| <b>ValueTextSize</b>         | Font size of the value display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <i>Double</i>                                                                                                                          |
| <b>ValueTextColor</b>        | Font size of the value display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <i>Color</i>                                                                                                                           |
| <b>IsGlasReflection</b>      | Activate the glass effect on the element.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <i>Boolean</i>                                                                                                                         |
| <b>GaugeOffsett</b>          | Lowering the rotation point of the whole element.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <i>Double</i>                                                                                                                          |

## 6.5.8 Diagramme Sankey

Le diagramme Sankey, WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).

L'assistant Sankey Wizard doit être utilisé pour modéliser un diagramme Sankey. L'assistant crée un fichier XML qui est ensuite évalué par l'élément WPF. Pour cela, la propriété **zSankeyName** doit recevoir le nom du fichier XML. Le fichier XML doit se trouver dans le dossier *Other* d'un projet. Il est enregistré ici par l'assistant.

Un exemple de diagramme Sankey dans le Runtime est illustré ci-dessous :



Les paramètres suivants peuvent être définis dans la fenêtre de configuration WPF sous **COPADATA-ELEMENT** :

| Propriété                   | Fonction                                                                                                                                                                               | Valeur                |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| <b>FontSize</b>             | Taille de la police de caractères des textes.                                                                                                                                          | <i>Integer</i>        |
| <b>zBackgroundColor</b>     | Couleur d'arrière-plan de l'élément.                                                                                                                                                   | <i>Color (String)</i> |
| <b>zFontColor</b>           | Couleur des textes.                                                                                                                                                                    | <i>Color (String)</i> |
| <b>zFontFamily</b>          | Police de caractères de tous les textes.                                                                                                                                               | <i>Font (String)</i>  |
| <b>zLossDetectionActive</b> | Détection automatique des pertes activée/désactivée. Si cette propriété est définie sur <i>true</i> , les pertes sont automatiquement affichées sous forme de flux au point d'un nœud. | <i>Bool</i>           |
| <b>zNoDataText</b>          | Texte affiché s'il n'existe aucune valeur à afficher et si <b>zPrevireActive</b> est définie sur                                                                                       | <i>String</i>         |

| Propriété                     | Fonction                                                                                                                                                                                                                                                                                                                      | Valeur         |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
|                               | <i>false</i> .                                                                                                                                                                                                                                                                                                                |                |
| <b>zNoValidXMLText</b>        | Texte affiché si aucun fichier valide avec un nom saisi n'a été trouvé et si <b>zPreviewActive</b> est définie sur <i>false</i> .                                                                                                                                                                                             | <i>String</i>  |
| <b>zNumberOfDecimalPlaces</b> | Indique le nombre de décimales devant être affichées.                                                                                                                                                                                                                                                                         | <i>Integer</i> |
| <b>zPreviewActive</b>         | Affichage de l'aperçu activé ou désactivé.<br><br>L'aperçu peut être affiché si :<br><br>Aucune donnée n'est présente (le diagramme modélisé est rempli avec des valeurs par défaut) ou<br><br>Le fichier XML est introuvable ou<br><br>Il ne contient pas de définition valide (un exemple de diagramme Sankey est affiché). | <i>Bool</i>    |
| <b>zRefreshRate</b>           | Fréquence d'actualisation du diagramme en ms.                                                                                                                                                                                                                                                                                 | <i>Integer</i> |
| <b>zSankeyName</b>            | Nom du fichier XML avec la modélisation du diagramme.                                                                                                                                                                                                                                                                         | <i>String</i>  |
| <b>zShowRelativeValues</b>    | Affichage des valeurs absolues avec l'état <i>false</i> ou des valeurs relatives avec l'état <i>true</i> .                                                                                                                                                                                                                    | <i>Bool</i>    |

**Remarque :** Une programmation supplémentaire dans VSTA est nécessaire pour l'affichage de diagrammes Sankey dans zenon Web Client. Vous trouverez des détails à ce sujet au chapitre Affichage d'éléments WPF dans zenon Web Client (à la page 110).

### 6.5.9 Temperature indicator - TemperatureIndicatorControl

| Property               | Function                                 | Value          |
|------------------------|------------------------------------------|----------------|
| <b>CurrentValue</b>    | Current value which should be displayed. | <i>Double</i>  |
| <b>MinValue</b>        | Minimum value of the scale.              | <i>Double</i>  |
| <b>MaxValue</b>        | Maximum value of the scale.              | <i>Double</i>  |
| <b>MajorTicksCount</b> | Number of main ticks on the scale.       | <i>Integer</i> |
| <b>MinorTicksCount</b> | Number of sub ticks on the scale.        | <i>Integer</i> |

| Property                      | Function                                                                         | Value                                                                                                                            |
|-------------------------------|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <b>TickNegativColor</b>       | Color of the negative main tick (gradient to TickPositivColor).                  | <i>Color</i>                                                                                                                     |
| <b>TickPositivColor</b>       | Color of the positive main tick (gradient to TickNegativColor).                  | <i>Color</i>                                                                                                                     |
| <b>MinorTickCount</b>         | Color of the sub ticks.                                                          | <i>Color</i>                                                                                                                     |
| <b>ElementBorderBrush</b>     | Color of the element border.                                                     | <i>Brush</i>                                                                                                                     |
| <b>ElementBackgroundBrush</b> | Color of element background.                                                     | <i>Brush</i>                                                                                                                     |
| <b>ElementGlasReflection</b>  | Activate the glass effect on the element.                                        | <i>Visibility</i>                                                                                                                |
| <b>ElementFontFamily</b>      | Element font.                                                                    | <i>Font</i>                                                                                                                      |
| <b>IndicatorColor</b>         | Color of the indicator fill color.                                               | <i>Color</i>                                                                                                                     |
| <b>IndicatorBorderColor</b>   | Color of the indicator border.                                                   | <i>Color</i>                                                                                                                     |
| <b>MajorTickSize</b>          | Size of main ticks on the scale.                                                 | <i>Size</i>                                                                                                                      |
| <b>MinorTickSize</b>          | Size of sub ticks on the scale.                                                  | <i>Size</i>                                                                                                                      |
| <b>ScaleLetteringDistance</b> | Distance of the scale caption (vertical), each x. main tick should be captioned. | <i>Integer</i>                                                                                                                   |
| <b>IndicatorScaleDistance</b> | Distance between indicator and scale (horizontal).                               | <i>Double</i>                                                                                                                    |
| <b>ScaleFontSize</b>          | Font size of the scale.                                                          | <i>Double</i>                                                                                                                    |
| <b>ScaleFontColor</b>         | Font color of the scale.                                                         | <i>Color</i>                                                                                                                     |
| <b>Unit</b>                   | Unit.                                                                            | <i>String</i>                                                                                                                    |
| <b>ElementStyle</b>           | Shape/type of element.                                                           | <i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>SmallFrame</i></li> <li>▶ <i>Unit</i></li> <li>▶ <i>None</i></li> </ul> |

## 6.5.10 Universal slider - UniversalReglerControl

| Property                      | Function                                         | Value                                                                                              |
|-------------------------------|--------------------------------------------------|----------------------------------------------------------------------------------------------------|
| <b>CurrentValue</b>           | Current value which should be displayed.         | <i>Double</i>                                                                                      |
| <b>ElementFontFamily</b>      | Element font.                                    | <i>Font</i>                                                                                        |
| <b>MinValue</b>               | Minimum value of the scale.                      | <i>Double</i>                                                                                      |
| <b>MaxValue</b>               | Maximum value of the scale.                      | <i>Double</i>                                                                                      |
| <b>Radius</b>                 |                                                  | <i>Double</i>                                                                                      |
| <b>ScaleRadius</b>            | Radius of the scale.                             | <i>Double</i>                                                                                      |
| <b>ScaleStartAngle</b>        | Angle at which the scale starts.                 | <i>Double</i>                                                                                      |
| <b>ScaleLabelRotationMode</b> | Alignment of the scale caption.                  | <i>Enum:</i><br>▶ <i>None</i><br>▶ <i>Automatic</i><br>▶ <i>SurroundIn</i><br>▶ <i>SurroundOut</i> |
| <b>ScaleSweepAngle</b>        | Angle area which defines the size of the scale.  | <i>Double</i>                                                                                      |
| <b>ScaleLabelFontSize</b>     | Font size of the scale caption.                  | <i>Double</i>                                                                                      |
| <b>ScaleLabelColor</b>        | Font color of the scale caption.                 | <i>Color</i>                                                                                       |
| <b>ScaleLabelRadius</b>       | Radius on which the scale caption is orientated. | <i>Double</i>                                                                                      |
| <b>ScaleValuePrecision</b>    | Accuracy of the scale caption.                   | <i>Integer</i>                                                                                     |
| <b>ElementStyle</b>           | Display type of the element                      | <i>Enum:</i><br>▶ <i>Knob</i><br>▶ <i>Plate</i><br>▶ <i>None</i>                                   |
| <b>MajorTickColor</b>         | Color of main ticks on the scale.                | <i>Color</i>                                                                                       |
| <b>MinorTickColor</b>         | Color of sub ticks on the scale.                 | <i>Color</i>                                                                                       |
| <b>MajorTickSize</b>          | Size of main ticks on the scale.                 | <i>Size</i>                                                                                        |
| <b>MinorTickSize</b>          | Size of sub ticks on the scale.                  | <i>Size</i>                                                                                        |

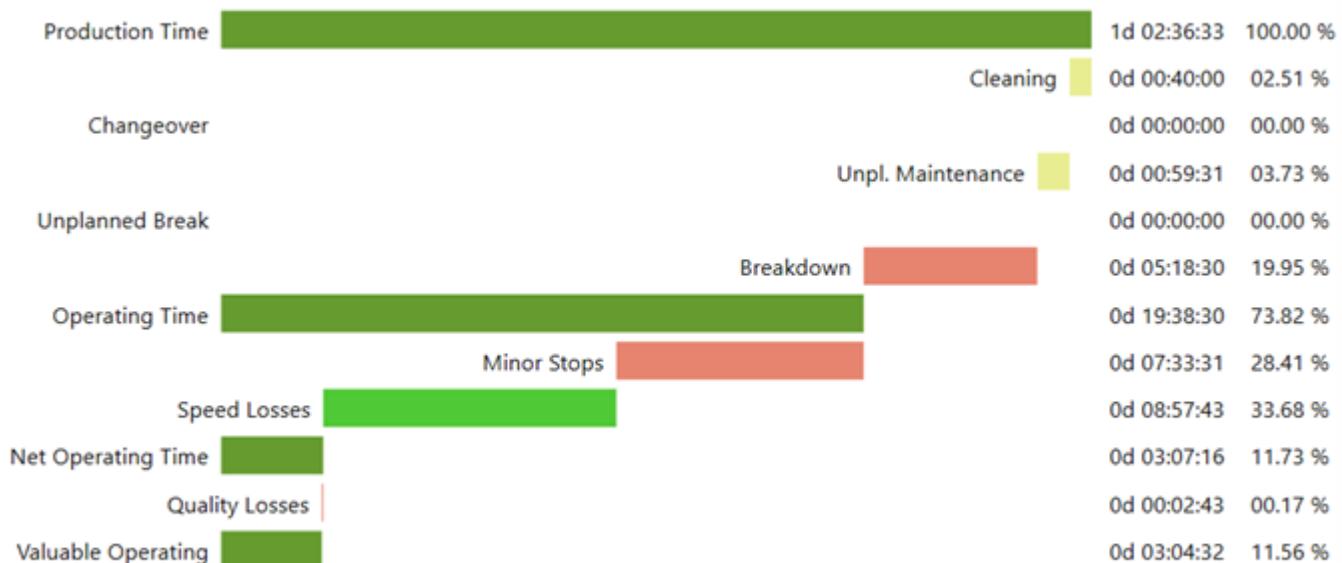
| Property                        | Function                                       | Value                                                    |
|---------------------------------|------------------------------------------------|----------------------------------------------------------|
| <b>MajorTicksCount</b>          | Number of main ticks on the scale.             | <i>Integer</i>                                           |
| <b>MajorTicksShape</b>          | Shape/type of main ticks on the scale.         | <i>Enum:</i><br>▶ Rectangle<br>▶ Trapezoid<br>▶ Triangle |
| <b>MinorTicksShape</b>          | Shape/type of sub ticks on the scale.          | <i>Enum:</i><br>▶ Rectangle<br>▶ Trapezoid<br>▶ Triangle |
| <b>MinorTicksCount</b>          | Number of sub ticks on the scale.              | <i>Integer</i>                                           |
| <b>BackgroundBorderBrush</b>    | Color of the element border.                   | <i>Brush</i>                                             |
| <b>BackgroundBrush</b>          | Color of element background.                   | <i>Brush</i>                                             |
| <b>PointerCapColorBrush</b>     | Color of pointer fastening point.              | <i>Brush</i>                                             |
| <b>GaugeMiddlePlate</b>         | Radius of the element background middle plate. | <i>Double</i>                                            |
| <b>ValueFontSize</b>            | Font size of the value display.                | <i>Double</i>                                            |
| <b>ValueFontColor</b>           | Font size of the value display.                | <i>Color</i>                                             |
| <b>IsGlasReflection</b>         | Activate the glass effect on the element.      | <i>Boolean</i>                                           |
| <b>KnobBrush</b>                | Color of the knob.                             | <i>Brush</i>                                             |
| <b>IndicatorBrush</b>           | Color of the indicator.                        | <i>Brush</i>                                             |
| <b>IndicatorBackgroundBrush</b> | Background color of the inactive indicator.    | <i>Brush</i>                                             |
| <b>KnobSize</b>                 | Diameter of the knob.                          | <i>Double</i>                                            |
| <b>KnobIndicatorSize</b>        | Indicator size of the knob.                    | <i>Size</i>                                              |
| <b>ElementSize</b>              | Size of the element.                           | <i>Size</i>                                              |
| <b>VisibilityKnob</b>           | Activating of the knob.                        | <i>Boolean</i>                                           |
| <b>ValuePosition</b>            | Position of the value display.                 | <i>Double</i>                                            |
| <b>ValueVisibility</b>          | Activating the value display.                  | <i>Boolean</i>                                           |

## 6.5.11 Diagramme en cascade

Le **diagramme cascade** WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).

L'assistant Meaning and Waterfall Chart Wizard doit être utilisé pour modéliser un diagramme en cascade. Un diagramme en cascade peut être modélisé avec cet assistant. Les informations sont enregistrées directement dans les variables dans la propriété **Paramètre pour le waterfall diagram** (groupe de propriétés de la variable **Analyzer**).

Un exemple de diagramme en cascade dans le Runtime est illustré ci-dessous :



**Note:** This screenshot is only available in English.

Les paramètres suivants peuvent être définis dans la fenêtre de configuration WPF sous **COPADATA-ELEMENT** :

| Propriété                       | Fonction                                            | Valeur         |
|---------------------------------|-----------------------------------------------------|----------------|
| <b>zenonRefreshRate</b>         | Délai entre les actualisations du diagramme, en ms. | <i>Integer</i> |
| <b>zenonWaterfallIdentifier</b> | Nom du diagramme en cascade.                        | <i>String</i>  |
| <b>zenonZSystemModel</b>        | Groupe d'équipements des                            | <i>String</i>  |

| Propriété | Fonction             | Valeur |
|-----------|----------------------|--------|
|           | variables utilisées. |        |

**Remarque :** Une programmation supplémentaire dans VSTA est nécessaire pour l'affichage de diagrammes en cascade dans zenon Web Client. Vous trouverez des détails à ce sujet au chapitre Affichage d'éléments WPF dans zenon Web Client (à la page 110).

## LIAISON DE BARRES À UNE FONCTION DE ZENON

Les barres d'un diagramme en cascade peuvent être liées à une fonction dans le Runtime. Pour l'affichage dans le Runtime, les barres, les intitulés et l'affichage des valeurs pour l'exécution avec un clic peuvent être configurés pour le diagramme en cascade.

Définissez la configuration suivante pour lier les colonnes de votre diagramme en cascade à une fonction :

1. Configurez l'élément WPF pour le diagramme en cascade.

**Remarque :** Pour cela, dans la mesure du possible, utilisez l'assistant Meaning and Waterfall Chart Wizard.

2. Développez une fonction dans zenon.

- a) Créez une nouvelle fonction :

Dans la barre d'outils ou le menu contextuel du nœud Fonctions, sélectionnez **Nouvelle fonction**.

La boîte de dialogue de sélection d'une fonction s'affiche à l'écran.

- b) Sélectionnez la fonction de votre choix.

- c) Configurez les paramètres pour la fonction.

3. Name the function in the **Nom** property.

**Remarque :** Le nom de la fonction doit contenir les variables du diagramme en cascade sans code de couleurs.

Vous trouverez également ces paramètres dans la propriété de la variable **Paramètre pour le waterfall diagram**, dans le groupe de propriétés **Analyzer**.

4. Liez la fonction au même groupe d'équipements que les variables.

**Remarque :** Vous trouverez cette liaison dans la propriété **Groupes d'équipements** de la fonction.

Les considérations suivantes s'appliquent à la configuration de ce projet :

- ▶ La fonction et les variables liées doivent être présentes dans le même projet zenon.
- ▶ Les variables doivent être liées à un groupe d'équipements.
- ▶ La fonction doit être liée au même groupe d'équipements que les variables.

## Exemple

Pour une barre avec la définition de cascade  $WF = WF1,02,05,\#E9ED92$ ; Le nom de la fonction (par exemple, *Function\_WF1,02,05*) doit être utilisé.

## 6.5.12 BACnet WPF Control

The **BACnet Schedule Control** WPF element is available to partners of COPA-DATA and is available to them via the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).

The control offers the possibility to configure BACnet schedule objects in a graphical user interface and to display these objects.

### 6.5.12.1 Installation

Requirements:

- ▶ You must be a registered COPA-DATA Partner.
- ▶ You need to have access to the COPA-DATA Partner Community already.

#### DOWNLOAD WPF ELEMENT FROM THE COPA-DATA PARTNER PORTAL

Carry out the following steps to download the WPF element:

1. Open the COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).
2. Log in in the area called **MY AREA**.
3. Enter "Partner" as a search term by the microscope.  
**COPA-DATA Partner Community** appears with a link in the results.
4. Click on the link.
5. In the **Partner Login Area**, click on **MORE**.

The **Overview** tab is opened.

6. Switch to the **Documents & Downloads** tab.
7. Under **SELECT CATEGORY**, activate the **WPF element** checkbox.
8. Click **Search**.

The labeling is hidden.

An orange frame surrounds the search field.

9. Click in the frame and enter **Bacnet**.
10. The **BACNetWPFScheduler** is shown.
11. Under **Actions**, click on the orange symbol to download the **BACnet Schedule Control**.
12. Save the folder locally.
13. Unzip the file.
14. Open the folder that contains the files *BacnetSchedulerControl.xaml* and *BacnetSchedulerWPF.dll*.

### IMPORT BACNET SCHEDULE CONTROL IN ZENON EDITOR

To create a WPF element

1. Open the zenon Editor
2. Go to the **Files** node in the project manager.
3. Expand the view of this node by clicking on the *[+]* button.  
The tree view is unfolded and the subnodes are displayed.
4. Go to the **Graphics** subnode.
  - a) Select the **Import file** entry in the tool bar of the detail view.  
The file selection dialog is opened.
  - b) Select the file called *BacnetSchedulerControl.xaml*.
  - c) Confirm the selection by clicking on the **Open** button
5. Go to the **Miscellaneous** subnode.
  - a) Select the **Import file** entry in the tool bar of the detail view.  
The file selection dialog is opened.
  - b) Select the file called *BacnetSchedulerWPF.dll*.
  - c) Confirm the selection by clicking on the **Open** button

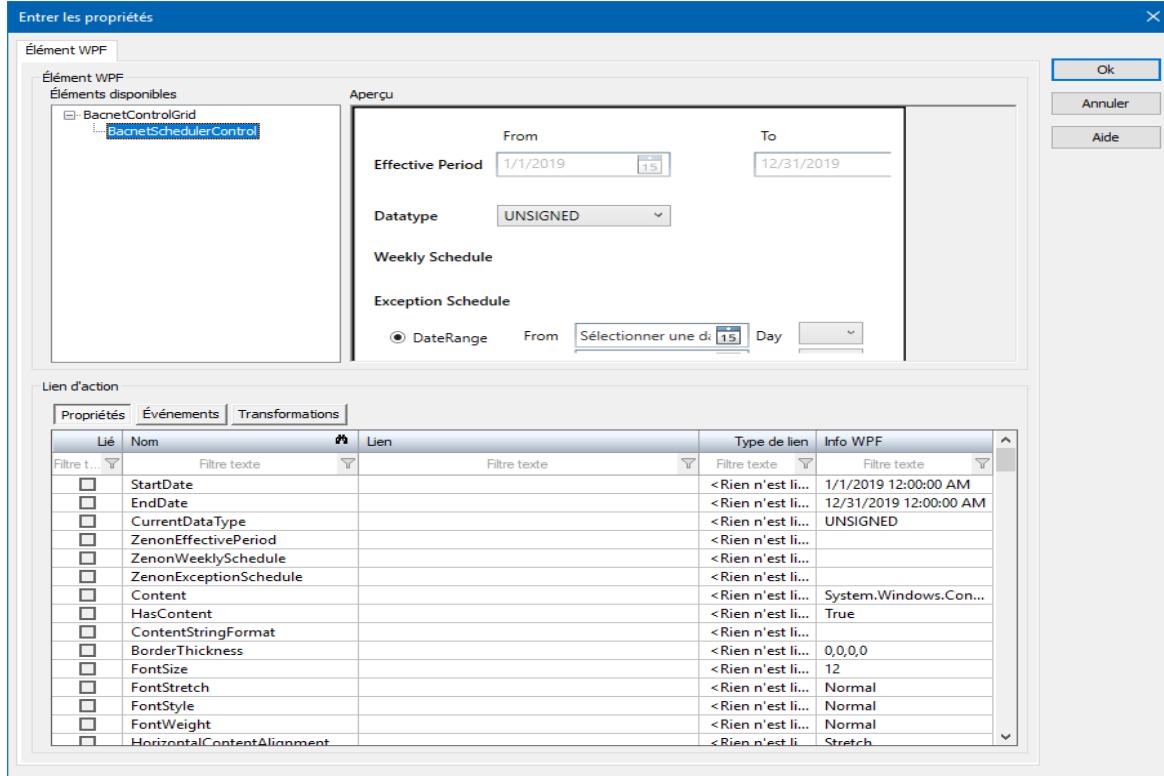
### 6.5.12.2 Engineering in the zenon Editor

Carry out the following steps to use the **BacnetScheduler** in zenon:

1. Create a new screen.  
To do this, select the **New screen** command in the tool bar or in the context menu of the **Screens** node.
2. Change the properties of the screen:

- a) Name the screen in the **Nom** property.
  - b) Select a desired screen type in the **Type de synoptique** property (recommended screen type: *Standard*).
  - c) Select the desired frame in the **Gabarit** property.
3. Configure the content of the screen:
- a) Open the screen in which the WPF element is to be inserted.
  - b) In the **Elements** tool bar, select WPF element (**WPF**) .
  - c) Drag the WPF element in the screen with the left mouse button.
  - d) The **File selection** window is opened.
  - e) Select the file **BacnetSchedulercontrol.xaml** and confirm the input by clicking on **OK**.
  - f) The WPF element is added.
4. Amend the size of the WPF element to your zenon screen.
- a) Highlight the desired WPF element.
  - b) Switch to the **Position** property group.
  - c) Set the parameters for the element width using the **Largeur [pixels]** property and the height with the **Hauteur [pixels]** property.  
Recommended minimum size: 920 x 920 pixels

### 6.5.12.3 Configurable control properties



Carry out the following steps to set the parameters for your BACnet WPF control:

1. Highlight the desired WPF element.
2. Switch to the **Liens WPF** property group.
3. Click the ... button in the **Configuration** property
4. The dialog to enter elements is opened.
5. Expand the **BacnetControlGrid** node in the **Existing Elements** area.  
The **BacnetSchedulerControl** is visible in the node.
6. Click on the **BacnetSchedulerControl** entry.  
The view in the **Action Linkings** area is updated.
7. Set the parameters for the WPF content:
  - a) Click once on the respective line in the **Linking Type** column.  
A drop-down list is opened.
  - b) Select the corresponding linking type in the drop-down list.
  - c) Click in the **Verknüpfung** column to open the **element input** dialog.  
This is amended to the configured linking type (for example, selection of a variable, entry of a value, ...)
8. Confirm your parameter settings by clicking on **OK**.

## POSSIBLE TYPES OF LINKING

One or more parameters can be linked for this WPF:

- ▶ *StartDate*
- ▶ *EndDate*
- ▶ *ZenonEffectivePeriod*
- ▶ *ZenonWeeklySchedule*
- ▶ *ZenonExceptionSchedule*

### Information

The setting of the parameters of the WPF element are visualized in zenon in the **Liens WPF** properties group of the WPF element. In doing so, each WPF linking is displayed with a separate subnode or property overview.

#### 6.5.12.4 Runtime view

From  To   Use Timespan

Effective Period

## Weekly Schedule

| Day | Time         | Value |
|-----|--------------|-------|
| Mon | 00:00:00.000 | [0]   |
| Tue | 00:00:00.000 | [0]   |
| Wed | 00:00:00.000 | [0]   |
| Thu | 00:00:00.000 | [0]   |
| Fri | 00:00:00.000 | [0]   |
| Sat | 00:00:00.000 | [0]   |
| Sun | 00:00:00.000 | [0]   |

## Exception Schedule

DateRange From   Day  Priority

SingleDate To   Day  Priority

WeekNDay

CalenderRef

## Time values

| Event Type        | Period                        | Priority |
|-------------------|-------------------------------|----------|
| DateRange         | 9/12/2018 - 9/14/2018         | 16       |
| SingleDate        | 9/15/2018                     | 16       |
| WeekNDay          | July;days numbered 1-7;Monday | 16       |
| CalenderReference | 0000001                       | 16       |

| Time         | Value     |
|--------------|-----------|
| 00:00:00.000 | [1] FALSE |

In zenon Runtime, existing BACnet schedule objects can be read by the PLC, modified in the Runtime and saved back to the PLC.

The user interface in Runtime is divided into three areas:

- ▶ **Effective Period**
- ▶ **Weekly Schedule**
- ▶ **Exception Schedule**

**Note:** The graphical user interface is only available in English.

#### 6.5.12.4.1 Runtime view

|                         |                                                                                                                         |                                                                                                                            |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| From                    | To                                                                                                                      | <input type="checkbox"/> Use Timespan                                                                                      |
| <b>Effective Period</b> | <input type="text" value="1/1/2018"/>  | <input type="text" value="12/31/2018"/>  |
|                         |                                                                                                                         | <b>Save</b>                                                                                                                |

In this area, the value of the variable that has been linked in configuration of the control (à la page 96) in the Editor for the **ZenonEffectivePeriod** option is displayed.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                         |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Use Timespan</b> | <p>The parameters for the Effective Period are set in Runtime. Input is in the input fields <b>From</b> and <b>To</b>.</p> <ul style="list-style-type: none"> <li>▶ <i>Inactive</i>:<br/>The Effective Period is not used.</li> <li>▶ <i>Active</i>:<br/>The Effective Period is entered in Runtime.</li> </ul> <p>Par défaut : <i>inactive</i></p> |
| <b>From</b>         | <p>Start of the Effective Period. Entry of the date. Manual entry or selection from a calendar. The calendar is displayed by clicking on the calendar symbol as a drop-down dialog.</p> <p>Par défaut : 1. 1. 2018</p> <p><b>Note:</b> Only active if the <b>Use Timespan</b> option is active.</p>                                                 |
| <b>To</b>           | <p>End of the Effective Period. Entry of the date. Manual entry or selection from a calendar. The calendar is displayed by clicking on the calendar symbol as a drop-down dialog.</p>                                                                                                                                                               |

| Parameter | Description                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------|
|           | Par défaut : 31. 12. 2018<br><b>Note:</b> Only active if the <b>Use Timespan</b> option is active. |
| Save      | Writes the current configuration for the <b>Effective Period</b> to the PLC.                       |

#### 6.5.12.4.2 Runtime view

**Weekly Schedule**

| Day | Time | Value |                                                                                               |
|-----|------|-------|-----------------------------------------------------------------------------------------------|
|     |      |       | <a href="#">Load</a><br><a href="#">New</a><br><a href="#">Delete</a><br><a href="#">Save</a> |

In this area, the value of each variable that has been linked in configuration of the control (à la page 96) in the Editor for the **ZenonWeeklySchedule** option is displayed.

| Parameters                                          | Description                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [List of the content of the loaded Weekly Schedule] | <p>The schedule is displayed in the <b>Weekly Schedule</b> list as a list entry. The parameters are set for the respective list entry:</p> <ul style="list-style-type: none"> <li>▶ <i>Day</i><br/>Selection of the day from a drop-down list.</li> <li>▶ <i>Time</i><br/>Entry of the time</li> </ul> |

| Parameters  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>Input format: <i>hh:mm:ss:ms</i></p> <ul style="list-style-type: none"> <li>▶ <i>Value</i><br/>Value of the respective schedule. The value represents the data type:</li> </ul> <p>[0] = <i>NULL</i><br/>     [1] = <i>BOOLEAN</i><br/>     [2] = <i>Unsigned</i><br/>     [3] = <i>REAL</i><br/>     [5] = <i>Double</i><br/>     [6] = <i>OctetString</i><br/>     [7] = <i>CharacterString</i><br/>     [8] = <i>BitString</i><br/>     [9] = <i>Enumerated</i><br/>     [10] = <i>Date</i><br/>     [11] = <i>Time</i><br/>     [12] = <i>BACnetObjectIdentifier</i></p> <p>Default: [0]</p> <p><b>Note:</b> Mandatory in the <i>Values</i> field when configuring the data type. The data type is given as a number according to the top list. Note that this must be given in square brackets [].<br/>Example: [1] for <i>BOOLEAN</i>.</p> <p>In addition to the data type, the corresponding value of the data type can also be configured. This value is entered after the data type.</p> <p>Examples:<br/>     [1] <i>TRUE</i><br/>     For Boolean with value <i>TRUE</i></p> <p>[10] <i>2/3/2021</i><br/>     For data type date with value February 3, 2021<br/>     Date entry in BACnet standard format.</p> |
| <b>Load</b> | <p>Reads the current configuration for the <b>Weekly Schedule</b> from the PLC.</p> <p>The loaded <i>Weekly Schedule</i> is displayed in the <b>Weekly Schedule</b> list.</p> <p><b>Note:</b> With a <b>Load</b>, all unsaved configurations are replaced with values from the PLC in Runtime</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

| Parameters    | Description                                                                                                                                                                                                                  |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | without a request for confirmation.                                                                                                                                                                                          |
| <b>New</b>    | Creates a new <b>Weekly Schedule</b> .                                                                                                                                                                                       |
| <b>Delete</b> | Deletes the selected entry from the <b>Weekly Schedule</b> list. The entry is deleted immediately without a request for confirmation.<br><br>Multiple selection is possible, but is ignored when clicking on <b>Delete</b> . |
| <b>Save</b>   | Writes the current configuration for the <b>Weekly Schedules</b> to the PLC.                                                                                                                                                 |

#### 6.5.12.4.3 Runtime view

##### Exception Schedule

DateRange      From      Select a date  Day      
   
 SingleDate      To      Select a date  Day  Priority 
  
 WeekNDay   
  
 CalenderRef

Time values

| Event Type | Period | Priority |
|------------|--------|----------|
|            |        |          |

| Time | Value |
|------|-------|
|      |       |

In this area, the value of the variable that has been linked in configuration of the control (à la page 96) in the Editor for the **ZenonExceptionSchedule** option is displayed.

| Parameter                                                                                                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type of Exception Schedule                                                                                  | <p>Type of schedule for exceptions (Exception Schedule).</p> <p>Selection from option box:</p> <ul style="list-style-type: none"> <li>▶ <i>DateRange</i><br/>The Exception Schedule is defined with a time period.<br/>The time period is configured with the options <b>From</b>, <b>Day</b>, <b>To</b> and <b>Day</b>.</li> <li>▶ <i>SingleDate</i><br/>The Exception Schedule is defined with a precise day.<br/>The day is configured with the options <b>Single</b> and <b>Day</b>.</li> <li>▶ <i>WeekNDay</i><br/>The Exception Schedule is defined with several days.<br/>These days are configured in the drop-down lists for <b>WeekNDay</b>.</li> <li>▶ <i>CalenderRef</i><br/>The Exception Schedule is defined with a reference to an existing BACNet calendar object.</li> </ul> |
| <b>From</b><br>(for Eventtype <i>DateRange</i> )<br><br><b>Single</b><br>(for Eventtype <i>SingleDate</i> ) | <p>Start of the Eventtype <i>Exception Schedule</i>. Entry of the date.</p> <p>Entry by means of manual entry or selection from a calendar. The calendar is shown by clicking on the calendar symbol as a drop-down dialog.</p> <p>Par défaut : <i>vide</i> (<i>Select a date</i>)</p> <p><b>Note:</b> Only active if the options <b>DateRange</b> or <b>SingleDate</b> are active.</p>                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Day</b><br>(in <b>From</b> line)                                                                         | <p>Weekday of the start of the Eventtype <i>Exception Schedule</i>.</p> <p>If a date is configured in the <b>From</b> option, the corresponding weekday is automatically set for the option.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| Parameter                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                   | <p>Selection of the weekday from a drop-down list:</p> <ul style="list-style-type: none"> <li>▶ <i>Mon</i><br/>Monday</li> <li>▶ <i>Tue</i><br/>Tuesday</li> <li>▶ <i>Wed</i><br/>Wednesday</li> <li>▶ <i>Thu</i><br/>Thursday</li> <li>▶ <i>Fri</i><br/>Friday</li> <li>▶ <i>Sat</i><br/>Saturday</li> <li>▶ <i>Sun</i><br/>Sunday</li> </ul> <p>Par défaut : <i>vide</i></p> <p><b>Note:</b> Only active if the options <b>DateRange</b> or <b>SingleDate</b> are active.</p> |
| <b>To</b>                         | <p>End of the Eventtype <i>Exception Schedule</i>. Entry of the date.</p> <p>Manual entry or selection from a calendar. The calendar is displayed by clicking on the calendar symbol as a drop-down dialog.</p> <p>Par défaut : <i>vide</i> (<i>Select a date</i>)</p> <p><b>Note:</b> Only active if the <b>DataRange</b> option is active.</p>                                                                                                                                |
| <b>Day</b><br>(in <b>To</b> line) | <p>Weekday of the end of the Eventtype <i>Exception Schedule</i>.</p> <p>If a date is configured in the <b>To</b> option, the corresponding weekday is automatically set for the option.</p> <p>Selection of the weekday from a drop-down list:</p> <ul style="list-style-type: none"> <li>▶ <i>Mon</i></li> </ul>                                                                                                                                                              |

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | <p>Monday</p> <ul style="list-style-type: none"> <li>▶ <i>Tue</i></li> <li>Tuesday</li> <li>▶ <i>Wed</i></li> <li>Wednesday</li> <li>▶ <i>Thu</i></li> <li>Thursday</li> <li>▶ <i>Fri</i></li> <li>Friday</li> <li>▶ <i>Sat</i></li> <li>Saturday</li> <li>▶ <i>Sun</i></li> <li>Sunday</li> </ul> <p><b>Note:</b> Only active if the <b>DataRange</b> option is active.</p>                                                                                                                                                                                                                      |
| <b>Priority</b>    | <p>Priority for the Eventtype in accordance with BACnet standard. Selection from drop-down list: Values 1 - 16</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>WeekNDay</b>    | <p>Selection of the date for the Eventtype <i>Exception Schedule</i> from drop-down lists.</p> <ul style="list-style-type: none"> <li>▶ Drop-down list 1:<br/>Selection of the month<br/><b>Note:</b> The <i>any month of year</i> entry corresponds to each month.</li> <li>▶ Drop-down list 2:<br/>Selection of the day in accordance with prescribed criteria.<br/>Drop-down list 3:<br/>Selection of the weekday<br/><b>Note:</b> The <i>any day of week</i> entry corresponds to each weekday of a week.</li> </ul> <p><b>Note:</b> Only active if the <b>WeekNDay</b> option is active.</p> |
| <b>CalenderRef</b> | <p>Reference to an existing BACnet <b>CalendarObject</b> object.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| Parameter                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                               | <p>Entry of the numerical <i>Object Instance</i> of the BACnet object. This is always a seven-digit bit number in accordance with the BACnet standard.</p> <p>Valid input range (in accordance with BACnet standard):<br/>0 - 4194302</p> <p><b>Note:</b><br/>To achieve this, the <i>Object Instance</i> number entered in the control must be filled with a 0 in front.</p> <p><b>Example:</b><br/><b>CalendarObject</b> <i>Instance Nummer</i> = 1<br/>Input in control = 0000001</p> |
| <b>[List of the content of the loaded Exception Schedule]</b> | <ul style="list-style-type: none"> <li>▶ The schedule is displayed in the <b>Exception Schedule</b> list as a list entry. The parameters are set with the listed options:</li> <li>▶ <i>Event Type</i>: Corresponds to the selection of the event types from the option box.</li> <li>▶ <i>Period</i>: Display of the date. The display can vary depending on the selected event type:<br/>[Date] - [Date]</li> <li>▶ <i>Priority</i>: Configured priority</li> </ul>                    |
| <b>Load</b>                                                   | <p>Reads the current configuration for the <b>Exception Schedules</b> on the PLC.</p> <p>The loaded <i>Exception Schedule</i> is shown in the list.</p> <p><b>Note:</b> With a <b>Load</b>, all unsaved configurations are replaced with values from the PLC in Runtime without a request for confirmation.</p>                                                                                                                                                                          |
| <b>New</b>                                                    | <p>Applies the configuration from the <b>Exception Schedule</b> area and creates a new entry in the list.</p>                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Delete</b>                                                 | <p>Deletes the selected entry from the <b>Exception Schedule</b> list. The entry is deleted immediately</p>                                                                                                                                                                                                                                                                                                                                                                              |

| Parameter   | Description                                                                                                            |
|-------------|------------------------------------------------------------------------------------------------------------------------|
|             | without a request for confirmation.<br>Multiple selection is possible, but is ignored when clicking on <b>Delete</b> . |
| <b>Save</b> | Writes the current configuration for the <b>Exception Schedules</b> to the PLC.                                        |

## TIME VALUES

Time values for a configured Exception Schedule Event.

| Parameter                                                                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [List of the configured Time values for the selected Exception Schedule Event] | <ul style="list-style-type: none"> <li>▶ <i>Time</i><br/>Entry of the time<br/>Input format: <i>hh:mm:ss:ms</i></li> <li>▶ <i>Value</i><br/>Value of the respective schedule. The value represents the data type:<br/> <ul style="list-style-type: none"> <li>[0] = <i>NULL</i></li> <li>[1] = <i>BOOLEAN</i></li> <li>[2] = <i>Unsigned</i></li> <li>[3] = <i>REAL</i></li> <li>[5] = <i>Double</i></li> <li>[6] = <i>OctetString</i></li> <li>[7] = <i>CharacterString</i></li> <li>[8] = <i>BitString</i></li> <li>[9] = <i>Enumerated</i></li> <li>[10] = <i>Date</i></li> <li>[11] = <i>Time</i></li> <li>[12] = <i>BACnetObjectIdentifier</i></li> </ul> </li> </ul> <p>Par défaut : [0]</p> <p><b>Note:</b> Mandatory in the <i>Values</i> field when configuring the data type. The data type is given as a number according to the top list. Note that this must be given in square brackets [].<br/>Example: [1] for <i>BOOLEAN</i>.</p> <p>In addition to the data type, the corresponding value of the data type can also be configured. This value is entered after the data type.</p> |

| Parameter | Description                                                                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           | <p>Examples:</p> <p>[1] TRUE<br/>For Boolean with value TRUE</p> <p>[10] 2/3/2021<br/>For data type date with value February 3, 2021<br/>Date entry in BACnet standard format.</p>                       |
| New       | Creates a new Time value entry for the selected Exception Schedule Event.                                                                                                                                |
| Delete    | <p>Deletes the selected Time value entry. The entry is deleted immediately without a request for confirmation.</p> <p>Multiple selection is possible, but is ignored when clicking on <b>Delete</b>.</p> |

### 6.5.12.5 Operation in zenon Runtime

Possibilities of the **BACnet Schedule Control** in zenon Runtime:

- ▶ Time period for validity of the schedule (*Effective Period*)
  - ▶ Configurable period of validity in days
  - ▶ Reading of the existing *Effective Period* from the PLC.
  - ▶ Saving of the modified *Effective Period* on the PLC.
- ▶ Configuration of weekly schedules:
  - ▶ Reading existing *Weekly Schedules* from the PLC.
  - ▶ Configuration of new entries for the *Weekly Schedule*
  - ▶ Deletion of existing entries
  - ▶ Saving of the modified *Weekly Schedules* to the PLC
  - ▶ Configuration possibility per entry:
    - day (Day of the Week)
    - time per day (local and UTC, depending on the PLC)
    - data type and value
- ▶ Configuration of exceptions (*Exception Schedule*)

- ▶ Reading of existing exception schedule from the PLC.
- ▶ Configuration of new entries for the Exception Schedule.
- ▶ Deletion of existing entries.
- ▶ Saving of the modified *Exception Schedule* to the PLC.
- ▶ Configuration possibility per entry:
  - Event Type (DateRange, SingleDate, WeekNDay, CalenderRef)
  - Period
  - Priority
  - Time
  - data type and value

## 6.6 Display of WPF elements in the zenon web client

In order to also be able to also use the pre-made WPF elements "**energy class diagram**", "**Sankey diagram**" and "**waterfall chart**" for the display in a zenon web client, amendments are necessary in the project:

- ▶ Engineering in the zenon Editor (à la page 110)
- ▶ Adapt VSTA code (à la page 111)

### 6.6.1 Engineering in the zenon Editor

Carry out the following project configuration steps in the zenon Editor, in order to also be able to display certain WPF elements in the zenon web client:

#### PLACE WPF IN THE ZENON SCREEN:

- ▶ Place the WPF element in a zenon screen.
- ▶ Give it a unique name in the **ID d'élément** property.  
You can find this property in the **Général** properties group.
- Note:** A warning dialog appears if the name for an element has already been issued in another screen.
- ▶ Use the element name issued here in the VSTA code.

## 6.6.2 VSTA code (complex)

In order to add the programmer code for the display of WPF elements in the zenon web client, carry out the following steps:

1. In the zenon Editor, switch to the **programmer interfaces** node.
2. Select the **VSTA** node and select the **Open VSTA Editor with project add-in...** with a right mouse click
3. The dialog to create a VSTA project is opened.
4. Select the C# entry in the **Create new VSTA project** dialog.
5. Create (copy) the code below.
6. Enter the name of the WPF element in the code.

**Note:** When opening the VSTA editor, note whether the content of the following code is already present in the project configuration. For the display of the WPF element in the web client, compare the existing code and undertake the necessary additions. Please note the comments in relation to this in the sample code.

### VSTA CODE

```
//As member:
zenOn.IDynPictures zScreens = null;
string[] WPFElements = {"WPF_Control", "WPFWebclient_1", "WPFWebclient_2"}; //Names of the WPF screen
elements that appear in the zenon project and that need access to the API (as many/few as you want)

//Add the following three lines of code in the project archive function:
void ThisProject_Active()
{
 zScreens = this.DynPictures();
 zScreens.Open += new zenOn.DDynPicturesEvents_OpenEventHandler(zScreens_Open);
 zScreens.Close += new zenOn.DDynPicturesEvents_CloseEventHandler(zScreens_Close);
}
//Add the following two lines of code in the project inactive function:
void ThisProject_Inactive()
{
 zScreens.Open -= new zenOn.DDynPicturesEvents_OpenEventHandler(zScreens_Open);
 zScreens.Close -= new zenOn.DDynPicturesEvents_CloseEventHandler(zScreens_Close);

//Final release and garbage collection of any API-Objects.
FreeObjects();
```

```
}

//Add two new event handlers:
void zScreens_Open(zenOn.IDynPicture obDynPicture)
{
foreach (string element in WPFElements)
{
if (obDynPicture.Elements().Item(element) != null)
{
obDynPicture.Elements().Item(element).set_WPFProperty("ELEMENT", "zenonVariableLink",
this.Variables().Item(0));
}
}
}

void zScreens_Close(zenOn.IDynPicture obDynPicture)
{
foreach (string element in WPFElements)
{
if (obDynPicture.Elements().Item(element) != null)
{
zenOn.IElement zWPFElement= obDynPicture.Elements().Item(element);
zWPFElement.set_WPFProperty("ELEMENT", "zenonTrigger", true);
zWPFElement = null;
}
}
}
```

### 6.6.3 VSTA code (simplified)

If only one WPF element is used in a zenon screen, the following more streamlined code can be used as an alternative. To do this, the names of the WPF element, and the screen in which the element is used, must be entered. This code is then recommended if, for each project, only one of the pre-made WPF elements is used.

#### VSTA CODE

```
zenOn.IDynPicture zScreen = zero;
string wpfElement = "WPF_Control"; //Name of the WPF element in the screen
string wpfPicture = "@Details_Overview_Online"; //Name of the zenon screen
```

```
//Add to the project active function:
void ThisProject_Active()
{
zScreen = this.DynPictures().Item(wpfPicture);
zScreen.Open += new zenOn.OpenEventHandler(zScreen_Open);
zScreen.Close += new zenOn.CloseEventHandler(zScreen_Close);
}

//Add to the project inactive function:
void ThisProject_Inactive()
{
zScreen.Open -= new zenOn.OpenEventHandler(zScreen_Open);
zScreen.Close -= new zenOn.CloseEventHandler(zScreen_Close);

//Final release and garbage collection of any API-Objects.
FreeObjects();
}

void zScreen_Open()
{
if (zScreen.Elements().Item(wpfElement) != null)
{
zScreen.Elements().Item(wpfElement).set_WPFProperty("ELEMENT", "zenonVariableLink",
this.Variables().Item(0));
}
}

void zScreen_Close()
{
if (zScreen.Elements().Item(wpfElement) != null)
{
zenOn.IElement zWPFElement = zScreen.Elements().Item(wpfElement);
zWPFElement.set_WPFProperty("ELEMENT", "zenonTrigger", true);
zWPFElement = null;
}
}
```

## 6.7 Examples: Integration of WPF in zenon

You can see how XAML files are created and integrated as WPF elements in zenon from the following examples:

- ▶ Integrate button as WPF XAML in zenon (à la page 118)
- ▶ Integrate bar graph as WPF XAML in zenon (à la page 114)
- ▶ Integrate DataGrid Control in zenon (à la page 124)

### 6.7.1 Integrate bar graph as WPF XAML in zenon

Example structure:

- ▶ Creating a bar graph (à la page 16) in Adobe Illustrator and converting it to WPF
- ▶ Integrate into zenon
- ▶ Linking with variables
- ▶ Adapting the bar graph WPF element

#### CREATE BAR GRAPH

The first step is to generate a bar graph as described in the Workflow with Adobe Illustrator (à la page 16) chapter. To be able to use the XAML file in zenon, insert this in the project tree in the **Files/graphics** folder.

#### INTEGRATE BAR GRAPH

**Note:** A zenon project with the following content is used for the following description:

- ▶ An empty screen as a start screen
- ▶ Four variables from the internal driver for
  - ▶ Scale 0
  - ▶ Scale central
  - ▶ Scale high
  - ▶ Current value
- ▶ A variable from the mathematics driver for displaying the current value (255)

To integrate the bar graph:

1. open the empty screen

2. place a **WPF element** (à la page 49) in the screen
3. select **Fichier XAML** in the properties window
4. Select the desired XAML file (for example **bar graph\_vertical.xaml**) and close the dialog

## ADJUST BAR GRAPH

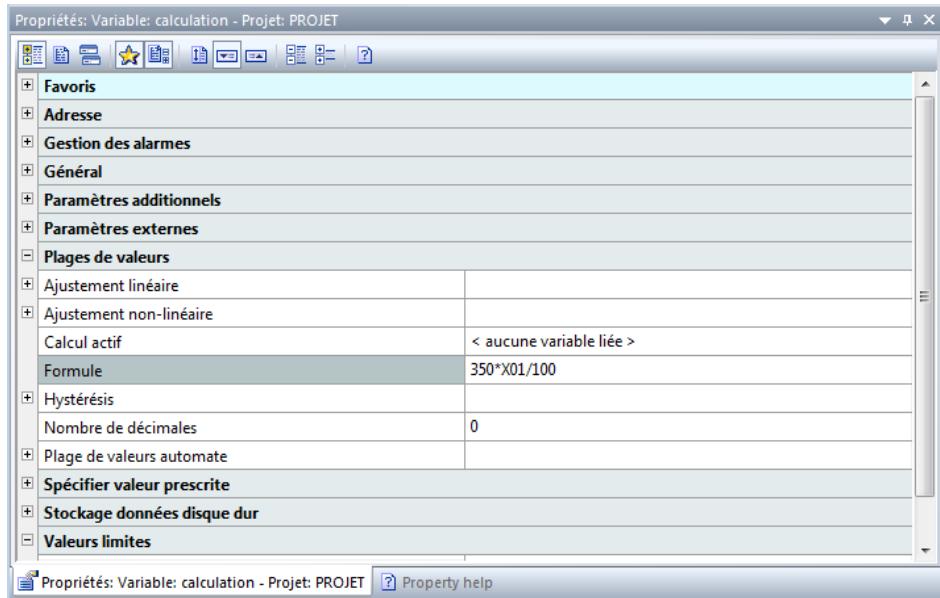
Before configuration, the scale of the XAML file is adapted if necessary:



To do this:

- ▶ Create a new mathematics variable that calculates the new value in relation to the scaling, for example:
  - ▶ Variable: 0-1000

- ▶ Mathematic variable {value created in xaml file}\*Variable/1000

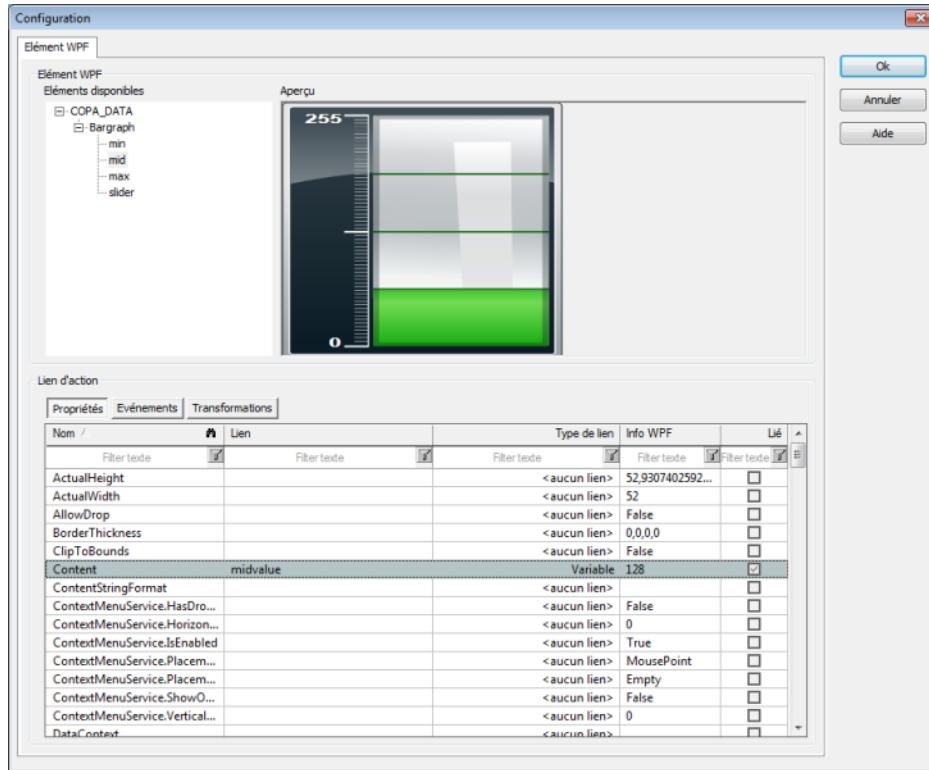


The XAML file is then configured.

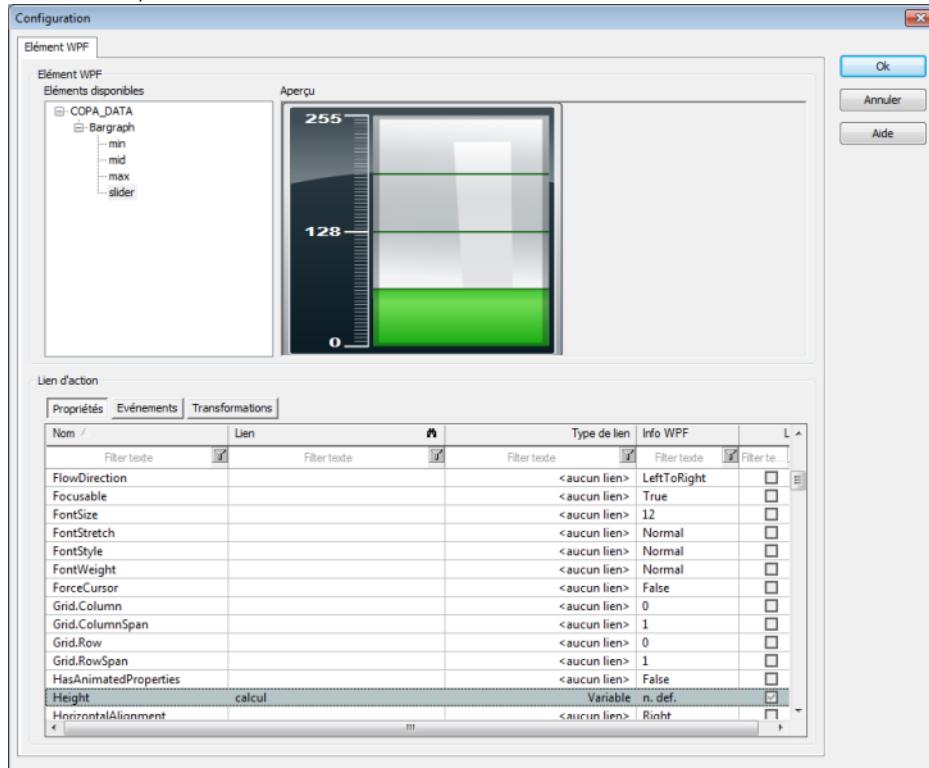
## CONFIGURE BAR GRAPH

1. Click on the WPF element and select the **Configuration** property
2. The configuration dialog shows a preview of the selected XAML file.

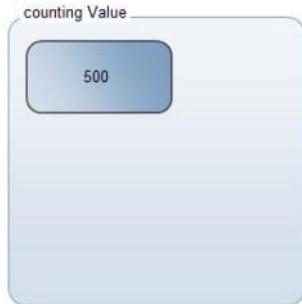
3. Select the minimum value, the average value and the maximum value and link each of these to the corresponding variable in the **Content** property



4. Select the **Slider** and link the **Value** property to the mathematics variables (in our example: calculation)



5. Check the project planning in Runtime:



### 6.7.2 Integrate button as WPF XAML in zenon

Example structure:

- ▶ Creating a button (à la page 12) in Microsoft Expression Blend
- ▶ Integrate into zenon
- ▶ Link to a variable and a function
- ▶ adjust the button to the size of the element
- ▶ Create button

As a first step, create a button as described in the Create button as XAML file with Microsoft Expression Blend (à la page 12) chapter. To be able to use the XAML file in zenon, insert this in the project tree in the **Files/Graphics** folder.

## INTEGRATE BUTTON

**Note:** A zenon project with the following content is used for the following description:

- ▶ An empty screen as a start screen
- ▶ an internal variable **int** of type **Int**

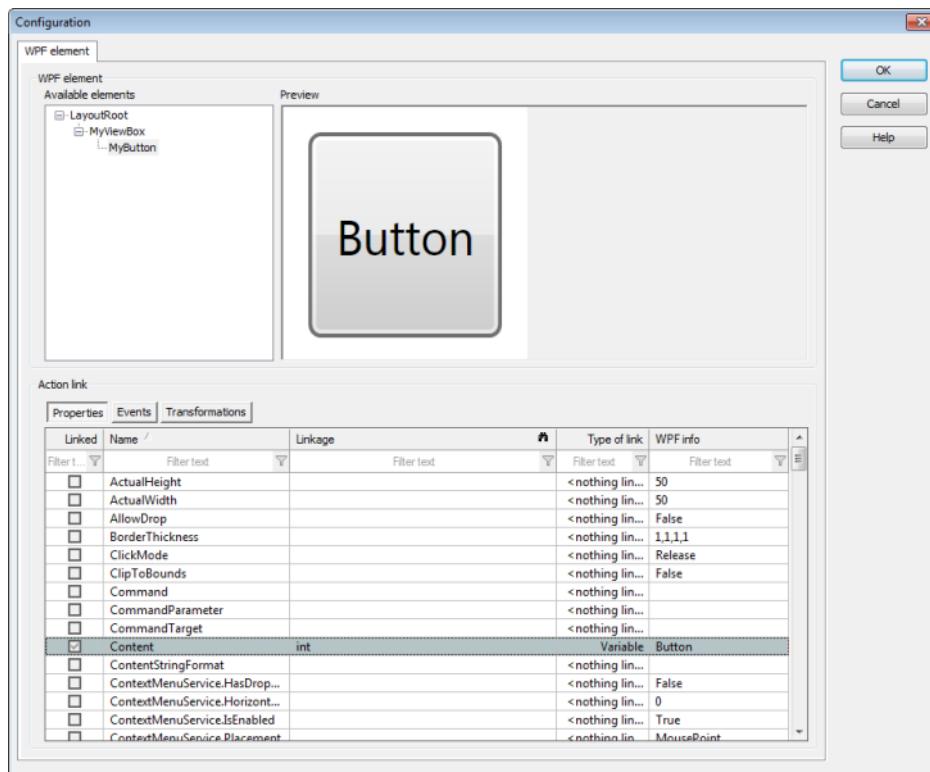
- ▶ A **Ecrire/modifier valeur prescrite Function\_0** function with:
  - ▶ **Direct to hardware** option activated
  - ▶ The target values should be configured with the value **45**.

To integrate the button:

1. open the empty screen
2. place a **WPF element** (à la page 49) in the screen
3. select **Fichier XAML** in the properties window
4. select the XAML file (e. g. **MyButton.xaml**) and close the dialog
5. select the **Configuration** property

## CONFIGURE THE BUTTON

The configuration dialog shows a preview of the selected XAML file. All elements named in the XAML file are listed in the tree:



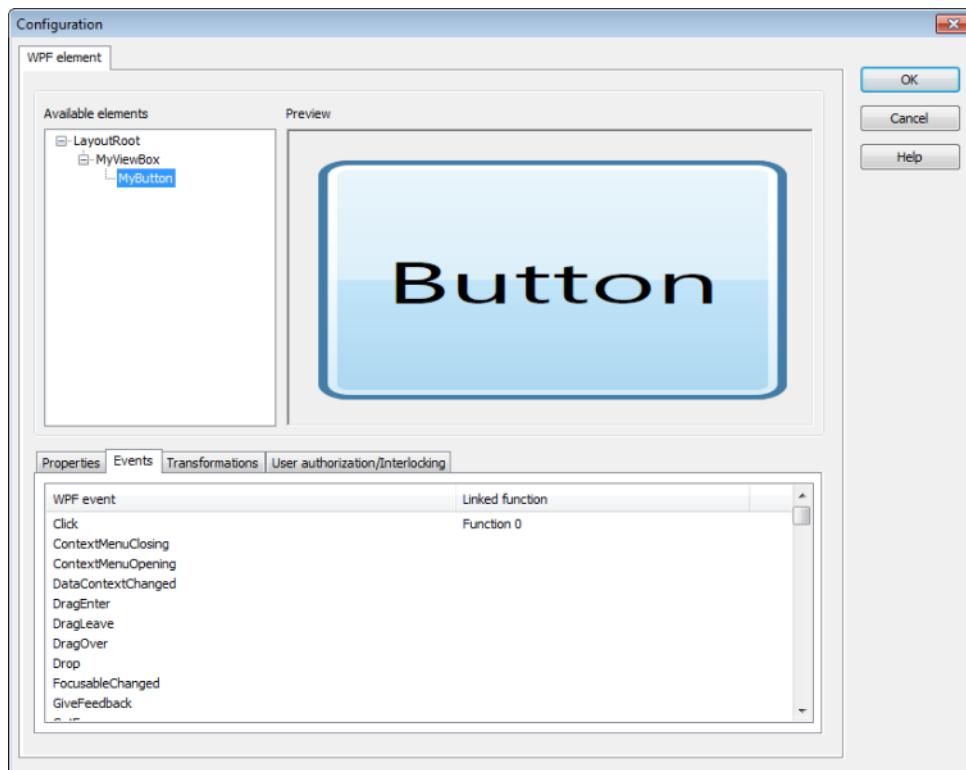
1. select the WPF button, which is in **LayoutRoot->MyViewBox->MyButton**
2. Look in the **Properties** Entry**Content** tab; this contains the button's text
3. Click the **Link type** column
4. Select **Variable** from the drop down list

5. Click in the **Link** column
6. the variable selection dialog is opened
7. select the *int* variable to link this variable with the **Content** property

## EVENTS

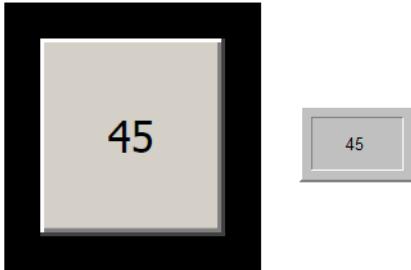
To also assign events:

1. select the tab Events



2. look for the 'Click' entry, this event is triggered by the WPF element, as soon as the button is clicked
3. Click in the **Link type** column
4. Select **Function** from the drop down list
5. Click in the **Link** column
6. the **function selection dialog** is opened
7. select **Function\_0**
8. Confirm the changes with **OK**
9. Insert a **numerical value element** into the screen
10. Link this **numerical value element** to the *int* variables too.
11. Compile the Runtime files and start Runtime.

The **WPF element** is displayed in Runtime, the button text is 0. As soon as you click on the button, the **click** event is triggered and the **set value** function is carried out. The value 45 is sent directly to the hardware and both **numerical value** and **button** display the value 45 .

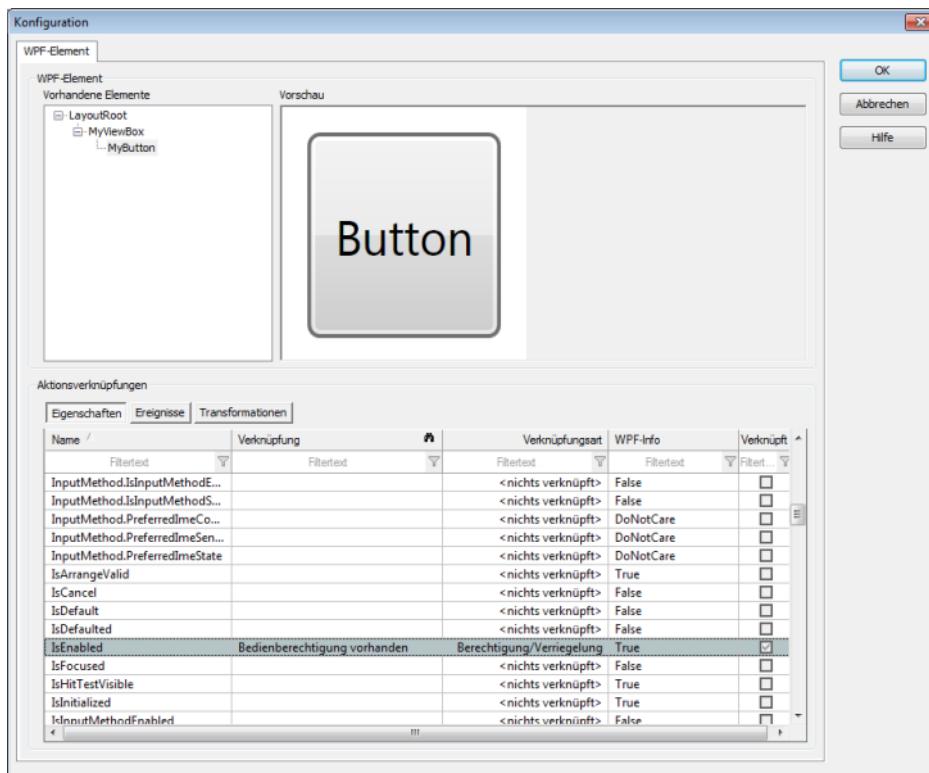


Define a set value of 30 via the **numerical value element**; this value is then also assumed by the **WPF element**.

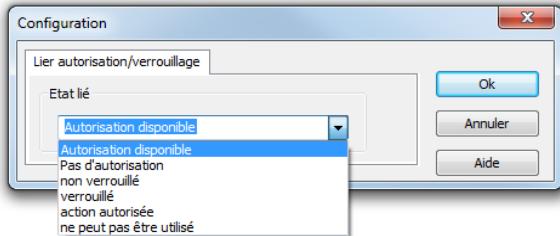
## AUTHORIZATION

Similar to a **numerical value**, a **WPF element** can be locked according to authorizations (lock symbol) or switched to be operable. Set the user authorization level to 1 for the **WPF element** and create a user called **Test** with **authorization level 1**. In addition, set up the functions **Login with dialog** and **Logout**. You link these two functions with 2 new text buttons on the screen.

In the **WPF element** configuration dialog, select the **MyButton** WPF button and select the **Properties:** tab

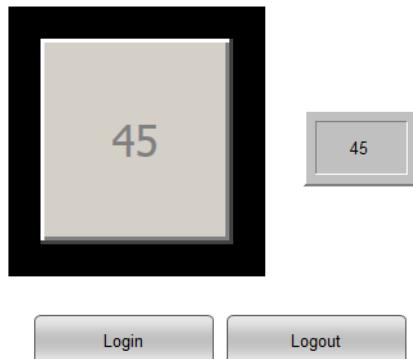


1. Select the **IsEnabled** element
2. Click in the **Link type** column
3. Select **Authorizations/interlocking** from the drop down list
4. Click in the **Link** column
5. In the drop-down list, select the *Authorized* option



6. Close the dialog with **OK**

Compile the Runtime file and note that Authorizations to be Transferred must also be selected. After Runtime has been started, the WPF button is displayed as deactivated on the screen and cannot be operated. If you now log in as the user **Test**, the button is activated and can be operated. The button is locked again as soon as you log out.



## TRANSFORMATION

The XAML files must still be adapted to use transformations:

1. switch to the **Expression Blend** program
2. select **MyButton**, so that the properties of the element are visible in the events window

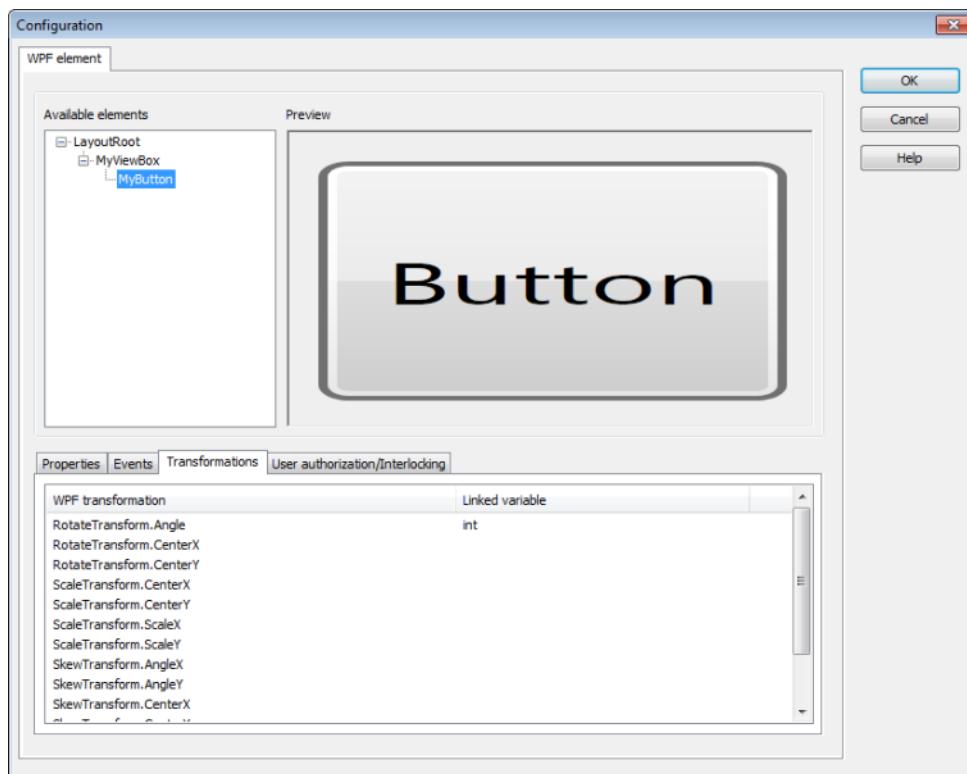


3. Under **Transform** at **RenderTransform** select the **Apply relative transform** option

As a result of this, a block is inserted into the XAML file, which save the transformation settings in runtime.

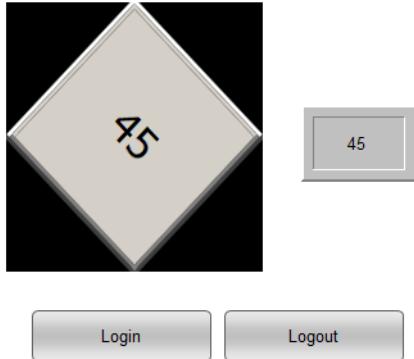
```
<Button.RenderTransform>
 <TransformGroup>
 <ScaleTransform ScaleX="1" ScaleY="1"/>
 <SkewTransform AngleX="0" AngleY="0"/>
 <RotateTransform Angle="0"/>
 <TranslateTransform X="0" Y="0"/>
 </TransformGroup>
</Button.RenderTransform>
```

4. Save the file and replace the old version in zenon with this new file.
5. Open the **WPF element** configuration dialog again:
  - a) select the **MyButton** button
  - b) select the **Transformations** tab



- c) select the element **RotateTransform.Angle**
- d) Click in the **Link type** column
- e) Select **Transformations** from the drop down list
- f) Click in the **Link** column
- g) the variable selection dialog is opened
- h) select the **int** variable to link this variable with the **RotateTransform.Angle** property

Compile the Runtime files and start Runtime. Log in as the **Test** user and click on the button. The button has the value **45** and the **WPF element** rotates by 45°.



### 6.7.3 Integrate DataGrid Control in zenon

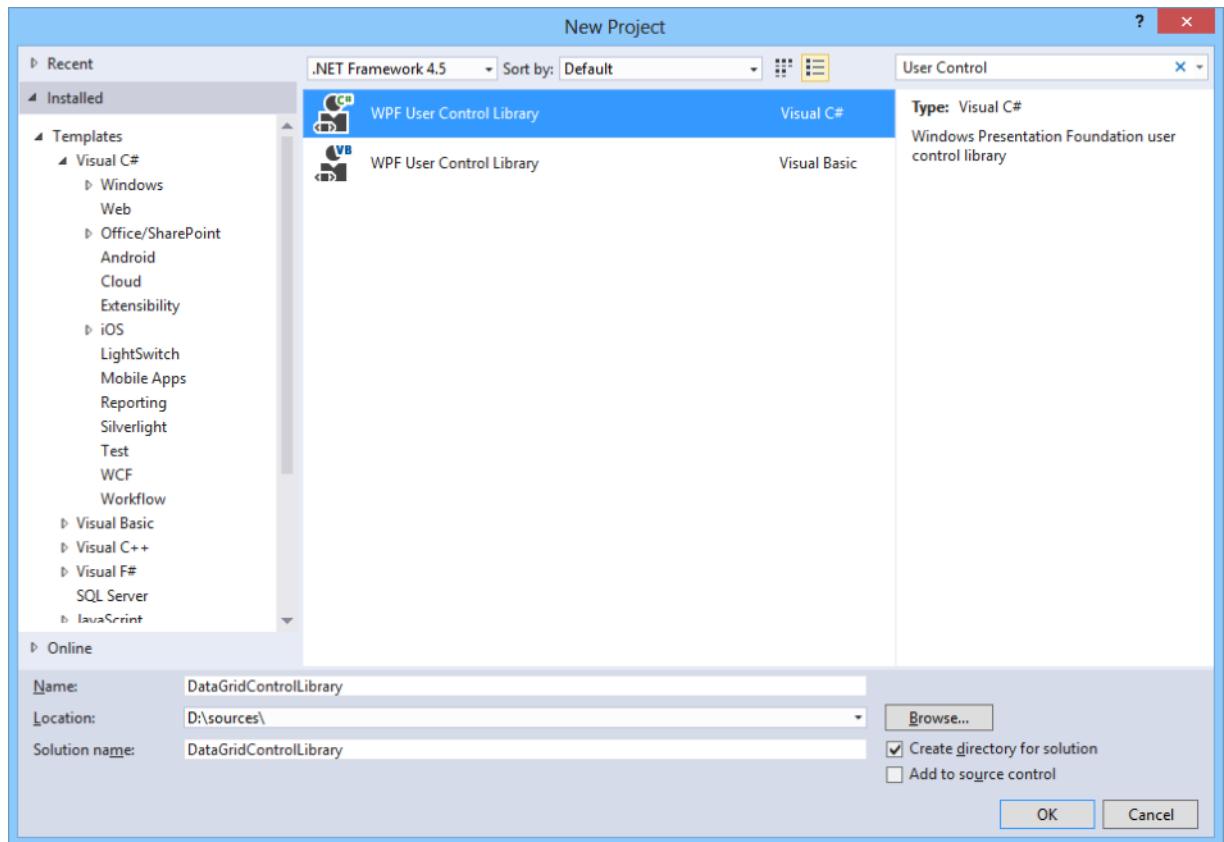
To create DataGrid control for zenon, you need:

- ▶ Visual Studio (Visual Studio 2015 in this example)

#### CREATE WPF USER CONTROL

1. in Visual Studio, create a new **Solution** and a **WPF User Control Library** project in .NET Framework version 4 or higher therein.

**Info:** If the corresponding project template does not appear in the list of available templates, this can be added by means of the search (field at the top right of the dialog).



In our example, the project is given the name **DataGridControlLibrary**.

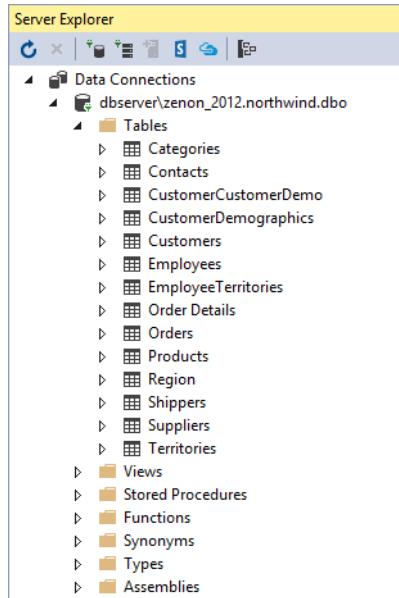
2. Create a new data connection in the **Server Explorer**.

In our example, the database *Northwind* is used, which is provided by Microsoft as an example database that can be downloaded for free.

To set up the database connection:

- a) Right-click on **Data Connections**.
- b) Select **Add connection....**
- c) Select *Microsoft SQL Server (SQLClient)* as **Data source**.
- d) Select the corresponding server and database name.

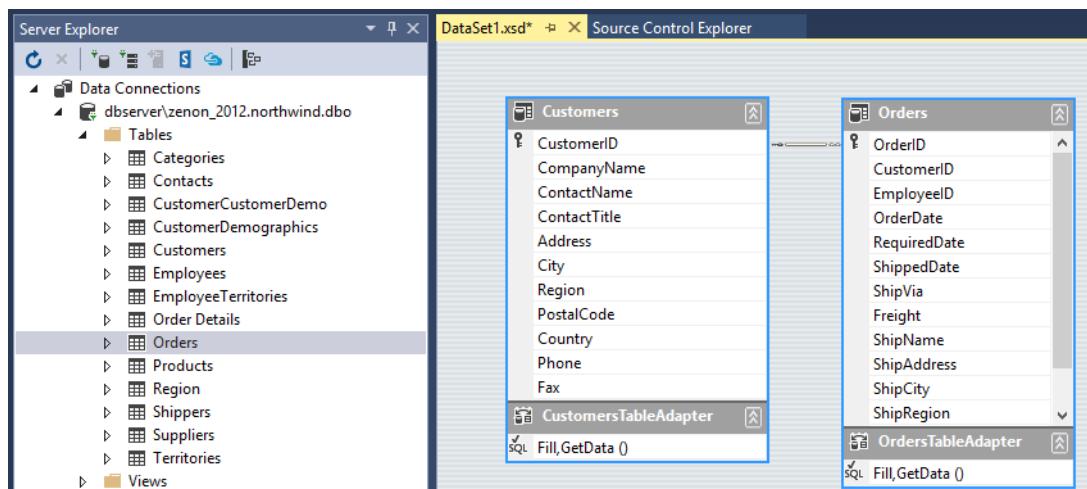
After adding the connection, the Server Explorer window should look a little like this:



A new DataSet is created in the next step.

## CREATING A DATASET

1. Right-click on the project
2. Select **Add – New Item...** in the context menu
3. Create a new **DataSet** with the name *DataSet1*.
4. Double click on the DataSet in order to open it in the Designer.
5. Drag the tables that you need (*Customers* and *Orders* in this example) to the DataSet design window.



The XAML file is modified in the next step.

## CONFIGURATION OF THE XAML FILE

1. If not already there, add the **Namespace** as a reference to the class in the XAML file:

```
<UserControl x:Class="DataGridControlLibrary.UserControl1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
 xmlns:local="clr-namespace:DataGridControlLibrary"
 mc:Ignorable="d"
 d:DesignHeight="300" d:DesignWidth="300">
```

2. Define the resources and the DataGrid that is to be used in the WPF:

```
<UserControl.Resources>
 <local:DataSet1 x:Key="DataSet1"/>
 <CollectionViewSource x:Key="CustomersViewSource" Source="{Binding Path=Customers,
 Source={StaticResource DataSet1}}"/>
</UserControl.Resources>
<Grid DataContext="{StaticResource CustomersViewSource}">
 <DataGrid Name="DataGrid1" DisplayMemberPath="CompanyName"
 ItemsSource="{Binding}" SelectedValuePath="CustomerID" HorizontalAlignment="Stretch"
 VerticalAlignment="Stretch"/>
</Grid>
```

3. Open the code-behind file (**UserControl1.xaml.cs**) and insert the following lines in the constructor:

```
public UserControl1()
{
 InitializeComponent();
 DataSet1 ds = ((DataSet1)(FindResource("DataSet1")));
 DataSet1TableAdapters.CustomersTableAdapter ta = new
 DataSet1TableAdapters.CustomersTableAdapter();
 ta.Fill(ds.Customers);
 CollectionViewSource CustomersViewSource =
 ((CollectionViewSource)(this.FindResource("CustomersViewSource")));
 CustomersViewSource.View.MoveCurrentToFirst();
}
```

In doing so, the following happens:

- ▶ The DataSet is obtained
- ▶ A new TableAdapter is created
- ▶ The DataSet is filled

- ▶ The information is provided to the DataGrid control

The solution can now be built.

## BUILD

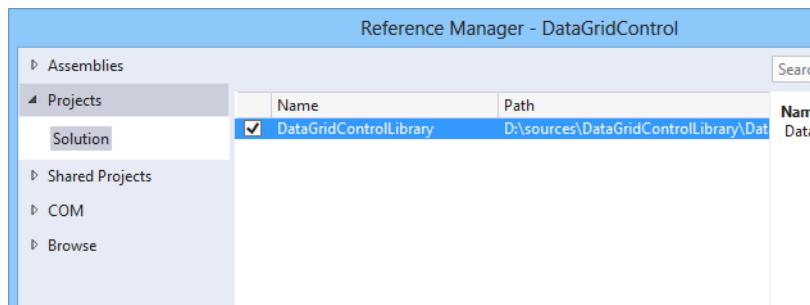
Now build the solution. The corresponding DLL (**DataGridColumnLibrary.dll**) is created in the output folder of the project.

Now you have a DLL with the necessary functionality available.

However zenon can only display XAML files that cannot be linked to the code behind file, which is why an additional XAML file is needed that references the DLL that has just been created.

To do this:

1. Create a further project, again as a **WPF User Control Library**
2. It was called **DataGridColumn** in our example.
3. Insert a reference to the project that has just been built into this new project.



4. The XAML files (**UserControl1.xaml**) looks as follows:

```
<UserControl x:Class="DataGridColumn.UserControl1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
 xmlns:local="clr-namespace:DataGridColumn"
 mc:Ignorable="d"
 d:DesignHeight="300" d:DesignWidth="300">
 <Grid>
 </Grid>
</UserControl>
```

5. Because all necessary content is contained in the DLL that has been created and no code-behind is necessary, delete the following lines:

```
x:Class="DataGridColumn.UserControl1"
xmlns:local="clr-namespace:DataGridColumn"
```

6. Also delete (for the positioning) the following lines:

```
mc:Ignorable="d"
d:DesignHeight="300" d:DesignWidth="300"
```

7. Delete the code-behind file (**UserControl1.xaml.cs**) in this project.
8. Define what is to be displayed in the XAML file.

To do this, modify the XAML file as follows:

```
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:dataGridLibrary="clr-namespace:DataGridControlLibrary;assembly=DataGridControlLibrary">

 <Grid x:Name="Grid1">
 <dataGridLibrary:UserControl1 Name="DataGridControl" HorizontalAlignment="Left"
 VerticalAlignment="Top"/>
 </Grid>
</UserControl>
```

The

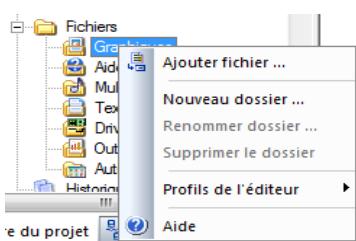
`lin xmlns:dataGridLibrary="clr-namespace:DataGridControlLibrary;assembly=DataGridControlLibrary"` defines the namespace **dataGridLibrary** and stipulates that this should use the assembly that has been created.

9. Assign a name for the grid.
10. Insert the control **dataGridLibrary:UserControl1** from our library and give it a name, because zenon can only modify objects that have a name.
11. Build the solution.

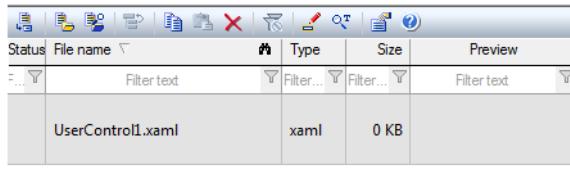
In the next step, how the DLL and XAML file are added to zenon is explained.

## STEPS IN ZENON

1. Open the zenon Editor
2. Go to *File -> Graphics*.
3. Select **Add file...** in the context menu



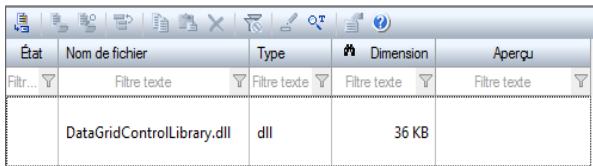
4. Select the XAML file at the save location (**UserControl1.xaml** from the **DataGridControl** project) and add this:



5. Insert the DLL with the functionality for the XAML file.

To do this:

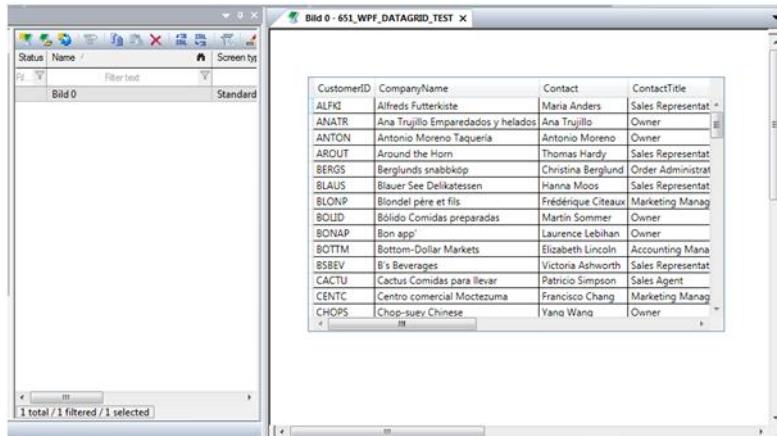
- Select, in the context menu, File -> Other**Add file....**
- Select the file **DataGridControlLibrary.dll** of the first project (**DataGridControlLibrary**).



6. Create a zenon screen.

7. Add a WPF element and select the previously-incorporated XAML file.

You should now see the following in the zenon Editor:



8. Start zenon Runtime in order to also test the control there.

## Hint |

DLLs that are part of a WPF element can also be replaced during ongoing operation. In doing so, the referencing is via linking in the XAML file.

To replace a DLL:

- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In Explorer, replace the DLL in the `\wpfcache` folder of the Editor files. You can find this folder in the SQL directory under  
`...\\PROJECT-GUID\\FILES\\zenon\\custom\\wpfcache`

As an alternative to replacement using Explorer, you can also replace the file in the zenon Editor directly. To do this, carry out the following steps:

- ▶ In the Visual Studio project settings, increase the file version of the DLL.
- ▶ Create the new DLL.
- ▶ Close all zenon screens in which the WPF element is used.
- ▶ Close all symbols that use a desired WPF element.
- ▶ In the zenon Editor, delete the DLL from the `\Files\Other` folder and add the file with the higher version number.

## 6.8 Error handling

### ENTRIES IN LOG FILES

Entry	Level	Meaning
<b>Xaml file found in %s with different name, using default!</b>	Warning	The name of the collective file and the name of the XAML file contained therein do not correspond. To avoid internal conflicts, the file with the name of the collective file and the suffix <b>.xaml</b> is used.
<b>no preview image found in %s</b>	Warning	The collective file does not contain a valid preview graphic ( <b>preview.png</b> or <b>[names of the XAML file].png</b> ). Thus no preview can be displayed.
<b>Xaml file in %s not found or not unique!</b>	Error	The collective file does not contain an XAML file or several files with the suffix <b>.xaml</b> . It cannot be used.
<b>Could not remove old</b>	Warning	There is an assembly that is to be replaced with a newer

Entry	Level	Meaning
<b>assembly %s</b>		version, but cannot be deleted.
<b>Could not copy new assembly %s</b>	Error	A new version is available for an assembly in the work folder, but it cannot be copied there. Possible reason: The old example is still loaded, for example. The old version continues to be used, the new version cannot be used,
<b>file exception in %s</b>	Error	A file error occurred when accessing a collective file.
<b>Generic exception in %s</b>	Error	A general error occurred when accessing a collective file.