



**zenon**  
by COPA-DATA



# Manuel de zenon Élément WPF de zenon

v.8.20



**COPADATA**

© 2020 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 ne sont utilisées que pour décrire le produit et ne sont pas des propriétés garanties au sens légal. Document sujet aux changements, techniques ou autres.

# Contenu

<b>1</b>	<b>Welcome to COPA-DATA help .....</b>	<b>5</b>
<b>2</b>	<b>WPF-Element.....</b>	<b>5</b>
<b>3</b>	<b>Notions fondamentales.....</b>	<b>6</b>
3.1	WPF in process visualization.....	7
3.2	Referenzierte Assemblies.....	8
3.3	Workflows.....	10
3.3.1	Workflow with Microsoft Expression Blend.....	11
3.3.2	Workflow mit Adobe Illustrator.....	11
<b>4</b>	<b>Leitfaden für Designer .....</b>	<b>12</b>
4.1	Workflow mit 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	Illustration Bargraf .....	16
4.2.2	WPF-Export.....	18
4.2.3	Animation in Blend.....	20
<b>5</b>	<b>Guidelines for developers .....</b>	<b>24</b>
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	Remplacement de données via VSTA .....	38
5.4	Zugriff auf das zenon (Runtime) Objektmodell aus einem WPF User Control.....	40
5.4.1	Zugriff über VSTA "Variablen-Link" .....	40
5.4.2	Access via marshaling .....	45
<b>6</b>	<b>Engineering in zenon .....</b>	<b>48</b>
6.1	Fichiers CDWPF (fichiers collectifs).....	48
6.2	WPF-Element anlegen.....	49
6.3	Konfiguration der Verknüpfung .....	50
6.3.1	Properties .....	52
6.3.2	Events.....	59
6.3.3	Transformation.....	60

6.4 Validité des fichiers XAML.....	63
6.5 Pre-built elements .....	65
6.5.1 Horloge analogique - AnalogClockControl.....	66
6.5.2 Bargraphe vertical - VerticalBargraphControl .....	66
6.5.3 Fortschrittsanzeige - ProgressBarControl .....	67
6.5.4 COMTRADE-Viewer.....	68
6.5.5 Energieklassen-Diagramm.....	81
6.5.6 Pareto-Diagramm .....	82
6.5.7 Circular gauge control .....	85
6.5.8 Sankey diagram .....	89
6.5.9 Temperaturanzeige - TemperatureIndicatorControl .....	90
6.5.10 Universal slider - UniversalReglerControl.....	92
6.5.11 Diagramme en cascade .....	94
6.5.12 BACnet WPF Control.....	96
6.6 Affichage d'éléments WPF dans zenon Web Client.....	112
6.6.1 Projektierung im zenon Editor.....	112
6.6.2 Code VSTA (complexe) .....	112
6.6.3 Code VSTA (simplifié) .....	114
6.7 Exemples : Intégration de WPF dans zenon .....	115
6.7.1 Intégration d'un bargraphe sous forme d'élément WPF XAML dans zenon .....	116
6.7.2 Intégration d'un bouton sous forme d'élément WPF XAML dans zenon.....	120
6.7.3 Integrate DataGrid Control in zenon.....	127
6.8 Error handling .....	133

## 1 Welcome to COPA-DATA help

### 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](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](mailto:documentation@copadata.com).

### ASSISTANCE PROJET

Vous pouvez obtenir de l'aide pour tout projet en contactant par e-mail notre service d'assistance : [support@copadata.com](mailto: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](mailto:sales@copadata.com).

## 2 WPF-Element

L'élément dynamique **WPF** permet d'intégrer et d'afficher les fichiers WPF/XAML valides dans zenon.

**Hinweis:** Im zenon Editor wird der Standard-Tooltip für das WPF-Element nicht angezeigt, wenn eine **.wpf** Datei verknüpft ist. Weiters wird in der zenon Runtime der zenon Tooltip für WPF Elemente nicht unterstützt.

## 💡 Informations

Alle Marken- und Produktnamen in dieser Dokumentation sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Titelhalter.

# 3 Notions fondamentales

## XAML

XAML est l'acronyme de **Extensible Application Markup Language**. Ce langage descriptif XML développé par Microsoft définit les éléments graphiques, les animations, les transformations, les affichages de dégradés de couleurs, etc. dans les interfaces utilisateur Silverlight et WPF. Le langage XAML permet de séparer de manière stricte la conception graphique et la programmation. Le concepteur prépare par exemple l'interface utilisateur graphique et crée des animations simples, qui sont ensuite utilisées par les développeurs ou ingénieurs de projet chargés de créer la logique de l'application.

## WPF

WPF est l'acronyme de **Windows Presentation Foundation**, et décrit une infrastructure graphique qui forme partie de l'infrastructure Windows .NET.

- ▶ WPF fournit un modèle complet aux programmeurs.
- ▶ Le langage XAML décrit, sur la base du langage XML, la hiérarchie de l'interface sous forme de langage de balisage. En fonction de la construction du fichier XAML, il est possible de lier des propriétés, des événements et des transformations d'éléments WPF à des variables et des fonctions de CD\_PRODUCTNAME<.
- ▶ Cette infrastructure réunit les différents éléments de présentation tels que l'interface utilisateur, les tracés d'éléments, les éléments graphiques, les éléments audio et vidéo, les documents et les polices de caractères.

Pour l'exécution dans zenon, Microsoft .NET framework version 4.6.2 ou une version ultérieure est requis.

## 💡 Informations

### Transparence

Pour que les contrôles WPF pour lesquels un arrière-plan transparent a été défini soient affichés comme transparents, les conditions suivantes doivent être remplies sur le poste de travail dans Editor et dans le Runtime :

- ▶ Le système d'exploitation doit être Windows 8.1 ou une version supérieure.
- ▶ .NET framework version 4.6.2 ou une version supérieure doit être installé.

Les éléments WPF ne sont pas affichés comme transparents dans Windows 7 ou 8. Au lieu de cela, les zones transparentes sont remplies avec la couleur d'arrière-plan définie dans le synoptique zenon.

## 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 Referenzierte Assemblies

Mittels den **WPF-Elementen** können nicht nur Standardobjekte (Rechtecke, Grafiken, etc.), bzw. Effekte (Farbverläufe, Animationen, etc.) dargestellt werden, sondern auch customisierte User Controls (mit Logik im Code Behind), die als Assemblies referenziert werden.

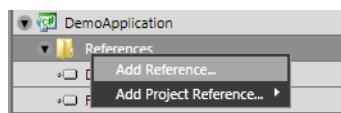
So kann zum Beispiel ein User Control erstellt werden, das wie ein Tacho aussieht und spezielle Eigenschaften und optische Effekte anbietet, etwa eine Eigenschaft **Value**, die beim Setzen dazu führt, dass der Zeiger des Tachos sich bewegt und/oder der entsprechende Wert in einer Beschriftung angezeigt wird.

Ablauf:

- ▶ Zeichnen Sie das Aussehen des User Controls mit Standard-Objekten, die von WPF angeboten werden.
- ▶ Programmieren Sie die Eigenschaften und Interaktionen.
- ▶ Kompilieren Sie das Gesamtpaket.

Das Ergebnis liegt als .NET Assembly vor.

Dieses Assembly kann für WPF-Projekte verwendet werden. Dafür muss es im WPF-Editor (z.B.: Microsoft Expression Blend) referenziert (verknüpft) werden. Wählen Sie dazu im zenon Datei-Auswahl dialog die Assembly aus:



Ab diesem Zeitpunkt können die WPF User Controls der Assembly in der Toolbox unter **Custom UserControls** ausgewählt und im WPF-Projekt verwendet werden.

Mehr dazu lesen Sie im Kapitel **Leitfaden für Entwickler** (à la page 24).

## REFERENZIERTE ASSEMBLIES IN ZENON VERWENDEN

Um ein Assembly in zenon zu verwenden, muss dieses als Datei zur Verfügung gestellt werden. Sammeldateien vom Format **.cdwpf** verwalten die Assemblies eigenständig. Es sind keine weiteren Konfigurationen nötig. Für Dateien vom Format **.xaml** müssen Assemblies zum Ordner *Dateien* hinzugefügt werden:

- ▶ Klicken Sie im Projektbaum auf *Dateien*.
  - ▶ Wählen Sie *Sonstige*.
  - ▶ Wählen Sie im Kontextmenü **Datei hinzufügen...**
- Der Dateiauswahl dialog wird geöffnet.
- ▶ Fügen Sie die gewünschte Assembly ein.

Beim Anzeigen einer WPF-Datei im **WPF-Element** (Editor und Runtime) werden die Assemblies aus diesem Ordner geladen. Damit ist auch sicher gestellt, dass beim Übertragen der Runtime-Dateien mittels **Remote-Transport** alle referenzierten Assemblies auf dem Zielrechner vorhanden sind.

Eine Sammeldatei (**.cdwpf**) und eine gleichnamige XAML-Datei können parallel existieren. Alle Assemblies (**\*.dll**) aus allen Sammeldateien und dem Ordner *Sonstige* werden in einen Arbeitsordner kopiert. Bei mehreren gleichnamigen Assemblies wird nur das mit der höchsten Dateiversion verwendet.

## 👉 Conseil

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.

## MEHRPROJEKTVERWALTUNG

In der Mehrprojektverwaltung muss in allen Projekten dieselbe Assembly verwendet werden. Wird in einem Projekt eine Assembly durch eine andere Version ersetzt, muss sie auch in allen anderen Projekten, die im Editor oder in der Runtime geladen sind, ersetzt werden.

### 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 120)

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 mit Adobe Illustrator

Aufbauend auf traditionelle Gestaltungsabläufe mit **Adobe Illustrator** bietet sich folgender Workflow an:

1. Illustration der Elemente in **Adobe Illustrator** (à la page 16)
2. Import der .ai Dateien und Aufbereitung in **MS Expression Design** (à la page 18)
3. WPF-Export aus **MS Expression Design** (à la page 18)
4. Animation in **MS Expression Blend** (à la page 20)
5. Integration in zenon (à la page 116)

Ein Beispiel zur Erstellung finden Sie im Kapitel Workflow mit Adobe Illustrator (à la page 16).

## 4 Leitfaden für Designer

Dieser Abschnitt informiert über die korrekte Erstellung von WPF Dateien im Microsoft Expression Blend und Adobe Illustrator. Die Tutorials zur Erstellung eines Button-Elements (à la page 12) und eines Bargraf-Elements (à la page 16) zeigen, wie in wenigen Schritten aus bereits existierenden Grafiken voll funktionsfähige WPF-Dateien für zenon erzeugt werden.

Folgende Werkzeuge werden dafür benutzt:

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

### Informations

Sollen referenzierte Objekte (Assemblies) in WPF verwendet werden, beachten Sie die Hinweise im Kapitel Referenzierte Objekte (à la page 8).

### 4.1 Workflow mit Microsoft Expression Blend

Mit Microsoft Expression Blend wird ein WPF-Element:

- ▶ illustriert
- ▶ über **MS Expression Design** in das WPF-Format umgewandelt
- ▶ animiert

Das folgende Beispiel zeigt die Illustration und Umwandlung eines Button-Elements in eine XAML-Datei.

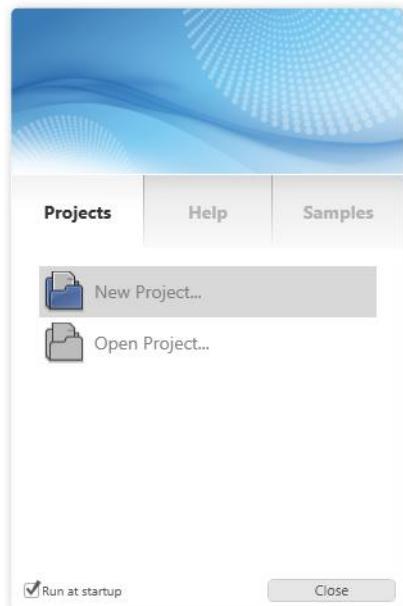
**Hinweis:** Eine Test-Version von „Microsoft Expression Blend“ steht auf der Microsoft Web-Seite zum Download zur Verfügung.

#### 4.1.1 Create button as an XAML file with Microsoft Expression Blend

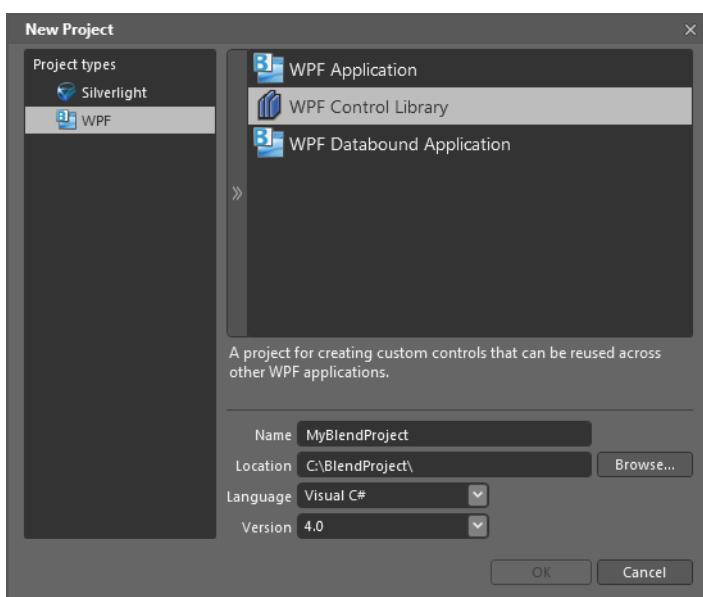
##### CREATE BUTTON

1. Start Expression Blend

2. select the **New Project** option



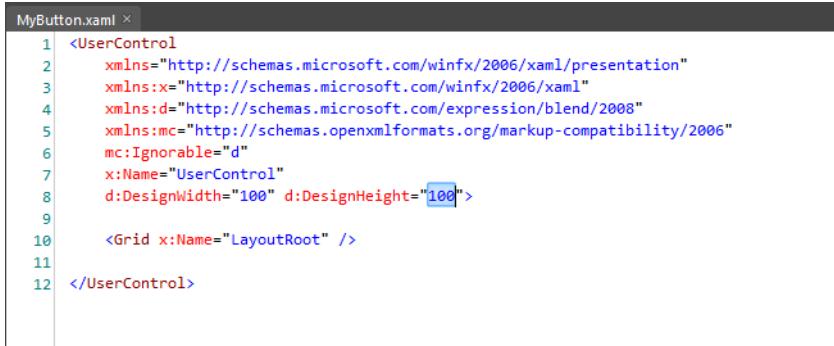
3. Select *WPF* as project type
4. give it a path and name of your choice (MyBlendProject, for example)



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

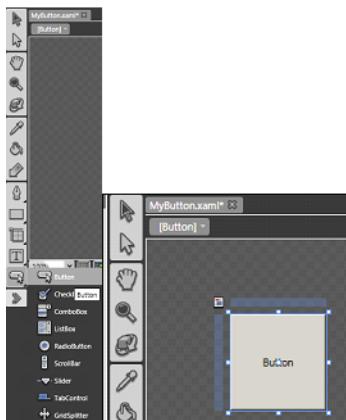
5. After the dialog has been confirmed with **OK**, Microsoft Blend creates a new project with the chosen settings. Expression Blend adds an empty XAML file which already contains a class reference.
6. Delete the CS file that belongs to the XAML file using the context menu.

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"`



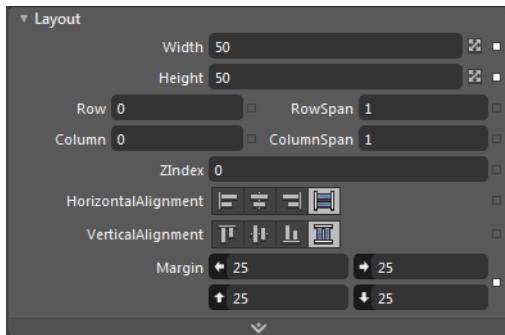
```
MyButton.xaml ×
1 <UserControl
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
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

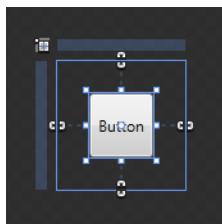


10. add a button via the toolbar
11. define the properties
  - ▶ Width: 50
  - ▶ Height: 50

- ▶ Margins: 25



The button is therefore at the center of the control.



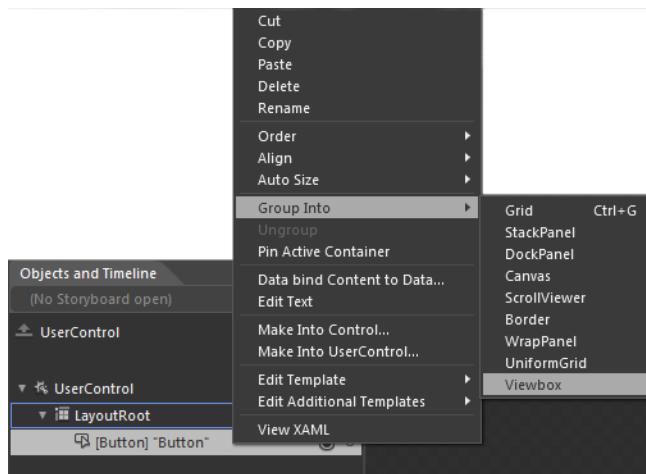
12. Save the changes and open the file in Internet Explorer to check it. You will see that the button is displayed in a size of 50 x 50 pixels.

## 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 120) 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 Illustration Bargraf

Ein Bargraf wird in Adobe Illustrator erstellt.

## 1. AI: Ausgangselement Bargraf



Illustriert in Adobe Illustrator CS3.

## 2. AI: Pfadansicht Bargraf in Adobe Illustrator



- ▶ alle Effekte müssen umgewandelt sein (**Objekt -> Aussehen umwandeln**)
- ▶ alle Linien werden in Pfade umgewandelt (**Objekt -> Pfad -> Konturlinie**)
- ▶ keine Filter wie Schlagschatten, Weichzeichner etc. verwenden

## HINWEISE ZUR KOMPATIBILITÄT

Illustrationen, die mit Adobe Illustrator erstellt werden, eignen sich grundsätzlich für den WPF-Export. Allerdings können nicht alle Illustrator-Effekte in Expression Design/Blend entsprechend werden.

Beachten Sie:

Effekt	Beschreibung
<b>Schnittmasken</b>	<p>In Adobe Illustrator erzeugte Schnittmasken werden von Expression Design nicht korrekt interpretiert. Meist werden sie in Blend als schwarze Farbflächen dargestellt.</p> <p>Wir empfehlen, die Illustrationen ohne Schnittmasken anzulegen.</p>
<b>Filter und Effekte</b>	<p>Nicht jeder Adobe Illustrator Filter wird in Expression Design entsprechend übernommen: So funktionieren z. B. Weichzeichnungsfilter, Schlagschatten sowie Eckeneffekte aus Illustrator in Expression Design nicht.</p> <p>Lösung:</p> <ul style="list-style-type: none"> <li>▶ Über den Befehl <b>Objekt -&gt; Aussehen Umwandeln</b> in Adobe Illustrator lassen sich die meisten Effekte so umwandeln, dass sie von Expression Design korrekt gelesen werden können.</li> <li>▶ Eckeneffekte aus Adobe Illustrator werden von MS Design korrekt interpretiert, wenn sie in AI in Pfade umgewandelt werden.</li> </ul>
<b>Textfelder</b>	Um Textfelder mit Code verknüpfen zu können, müssen diese in

Effekt	Beschreibung
	<p>Expression Blend gesondert angelegt werden. „<b>Labels</b>“ werden für dynamische Texte benötigt, für statische Informationen genügen einfache „<b>Textfelder</b>“.</p> <p>Die Möglichkeit, Textlabels zu erstellen, wird in MS Design nicht angeboten. Diese müssen direkt in MS Blend erstellt werden.</p>
<b>Transparenzen und Gruppentransparenzen</b>	<p>Zu Schwierigkeiten kann es bei der korrekten Interpretation von Transparenz-Einstellungen, insbesondere von Gruppentransparenz-Einstellungen, in Adobe Illustrator kommen.</p> <p>MS Expression Blend sowie MS Expression Design bieten aber die Möglichkeit, Transparenz-Einstellungen neu anzulegen.</p>
<b>Ebenen multiplizieren</b>	<p>Diese Ebenen-Einstellungen in Adobe Illustrator werden von MS Expression Blend nicht immer richtig dargestellt.</p> <p>Es gibt aber die Möglichkeit, „<b>Ebene multiplizieren</b>“ direkt in Expression Design einzustellen.</p>
<b>Zeigerinstrumente und Standardpositionen</b>	<p>Um die Grafiken optimal für die Animation vorzubereiten, müssen Zeiger und Regler immer auf der Ausgangsposition, meist 0 oder 12:00 Uhr stehen.</p> <p>So werden die Positions-Parameter für Rotationen etc. gleich richtig in Blend angegeben und eine Animation kann ohne Umrechnung von Positionsangaben erfolgen.</p>

## 4.2.2 WPF-Export

Für die Animation in Microsoft Expression Blend werden WPF-Dateien benötigt. Wir empfehlen Microsoft Expression Design für diesen Export, da es gute Ergebnisse liefert und die meisten Illustrator Effekte korrekt interpretiert werden.

**Hinweis:** Im Internet wird ein gratis Plug-in für den direkten Export von WPF-Dateien aus Adobe Illustrator angeboten. Dieses Plug-in bietet einen schnellen, unkomplizierten Export aus Illustrator, ist für die aktuelle Anwendung jedoch weniger gut geeignet, da es zu grafischen Verlusten kommt. Auch Farbabweichungen vom ursprünglichen Dokument sind möglich.

Dateien im Format **.ai** können regulär in Expression Design importiert werden, die Pfade bleiben dabei erhalten.

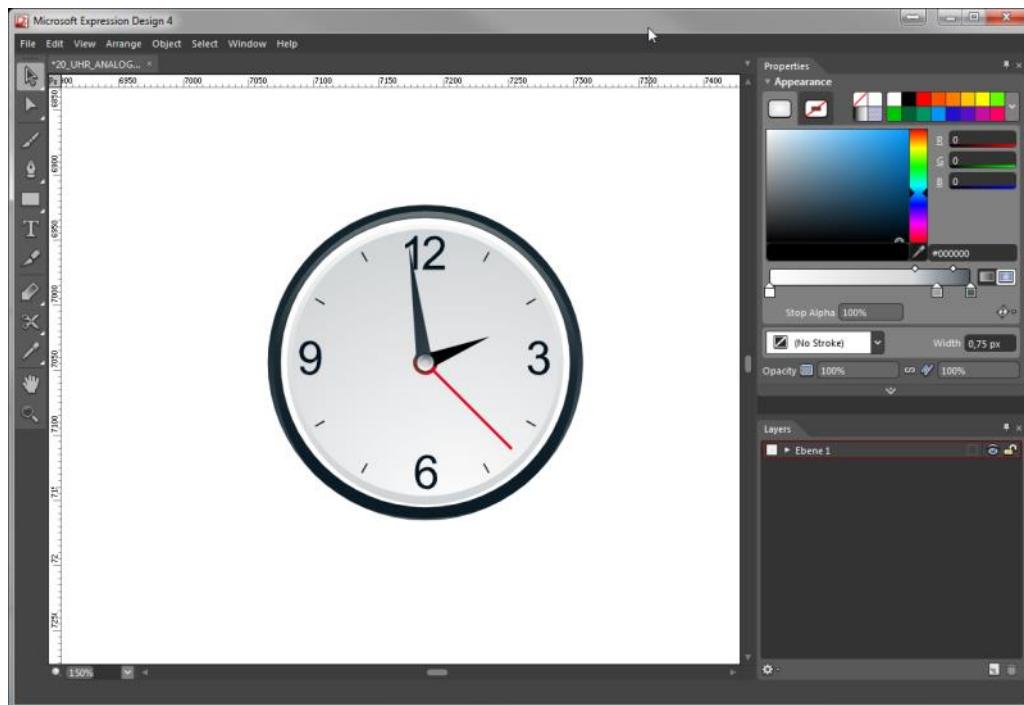
**Achtung:** Einige geläufige Illustrator Effekte können jedoch von Expression Design nicht richtig dargestellt werden (siehe Kapitel Illustration (à la page 16)).

In 5 Schritten exportieren wir das bereits erstellte Bargraf-Element:

## 1. **ED: Import**

- ▶ importieren Sie die aufbereitete Illustrator-Datei (à la page 16) in **Microsoft Expression Design** über **Datei -> Importieren**

## 2. **ED: Optimierung**



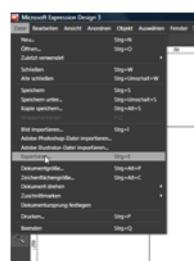
- ▶ wird die Ausgangsdatei in MS Expression Design nicht richtig angezeigt, kann sie hier noch nachbearbeitet und optimiert werden

## 3. **ED: Auswahl**



- ▶ markieren Sie mit dem **Direktauswahl**-Pfeil in MS Expression Design das Element für den WPF-Export, in diesem Fall die gesamte Uhr

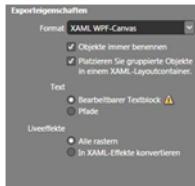
## 4. **ED: Export starten**



- ▶ starten Sie den Export über **Datei -> Exportieren**

- ▶ der Dialog zur Konfiguration der Exporteinstellungen öffnet sich

## 5. **ED: Exporteinstellungen**



- ▶ Nehmen Sie folgende Exporteinstellungen vor:

a) **Format:** *XAML Silverlight 4 / WPF Canvas*

**Objekte immer benennen:** Mit Häkchen aktivieren

**Platzieren Sie gruppierte Objekte in einem XAML Layoutcontainer:** Mit Häkchen aktivieren

b) **Text:** *Bearbeitbarer Textblock*

c) **Linieneffekte:** *Alle rastern*

Die exportierte Datei hat die Dateiendung **.xaml**. Sie wird im nächsten Schritt in MS Expression Blend aufbereitet und animiert (à la page 20).

### 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

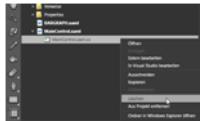
In thirteen steps, we go from a static XAML to an animated element, that can be embedded in zenon:

## 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 a framework that is supported by zenon

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



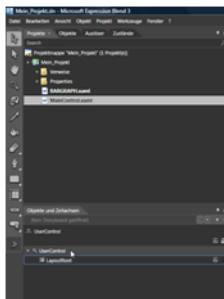
- a) Navigate to **MainControl.xaml.cs**
- b) Delete this file using the **Delete** command in the context menu

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



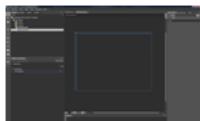
- a) Open the context menu for **My\_Project** (right mouse button)
- b) Select **Add existing element...**
- c) Select the XAML file exported from Microsoft Expression Design, in order to open this in Microsoft Expression Blend

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



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

## 5. **EB: Adapt XAML code**



- a) Click on **UserControl** with the right mouse button
- b) Select **Display XAML** in the contextual menu.
- c) 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



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

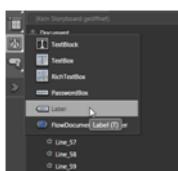
## 8. EB: Insert element



- a) Click on **Layout Root** with the right mouse button
- b) 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**

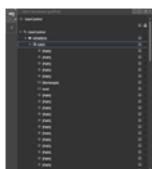
- ▶ Dynamic and static texts are labeled with text fields
- ▶ Values (numbers) are issued with **Labels**

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 116)

**A hint 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 Guidelines for developers

This section handles the creation of simple WPF user controls with code-behind functionality using Microsoft Visual Studio and debugging this user control in the Runtime.

The following tools were used for this:

- ▶ Microsoft Visual Studio 2015
- ▶ zenon

### Informations

Recommendation: Use Microsoft Visual Studio version 2012 or later. The XAML designer is better integrated in this.

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

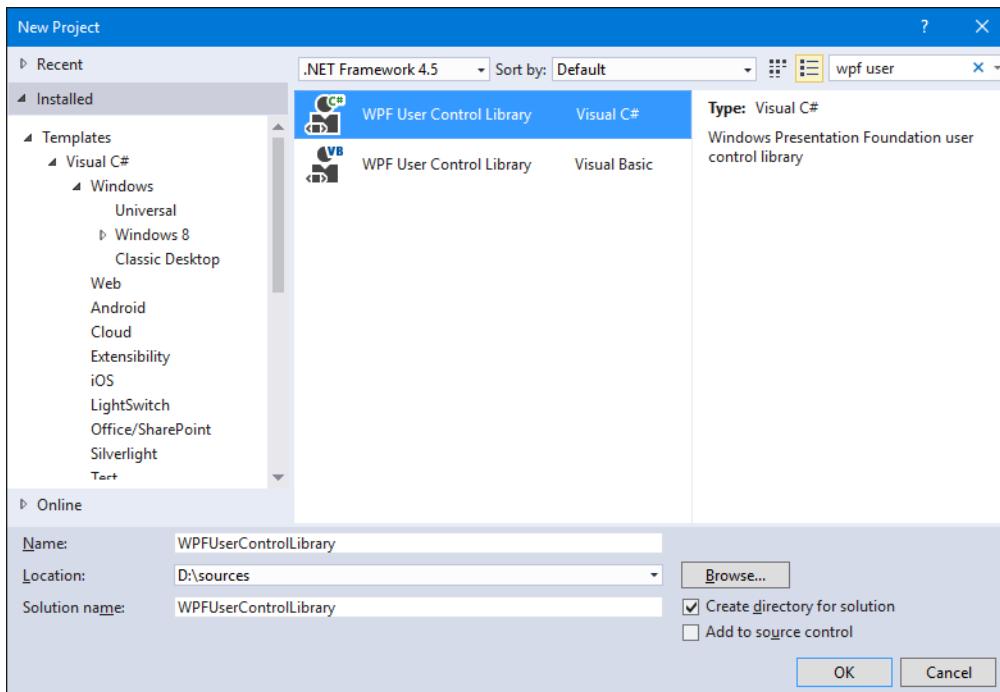
The creation and incorporation of a simple user control is described in this chapter. Because only the fundamental mechanisms/process for integration into zenon is described, the functionality of the user control is limited to the addition of two values. There is intentionally no enhanced error handling 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:id="http://schemas.microsoft.com/expression/blend/2008"
    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;
    }
}

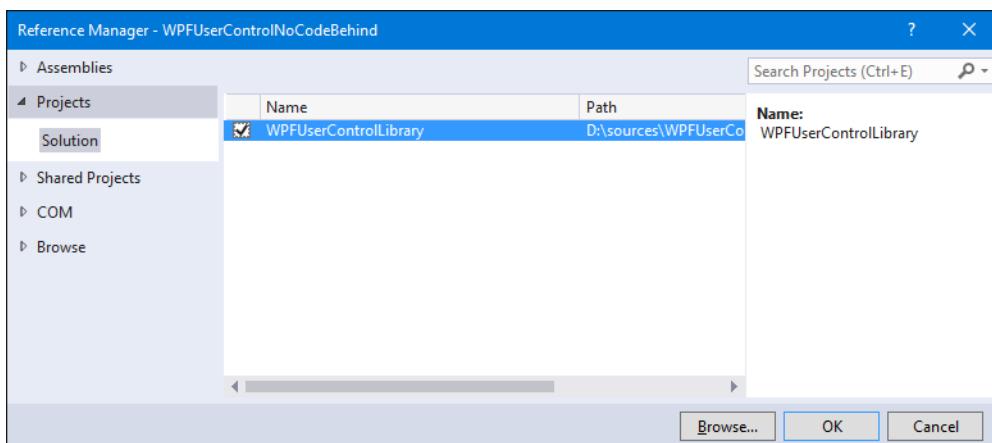
```

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.

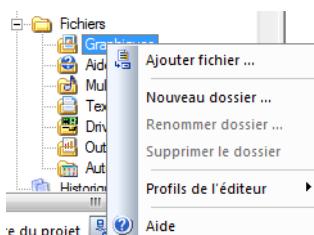
10. The XAML file should now look as follows:



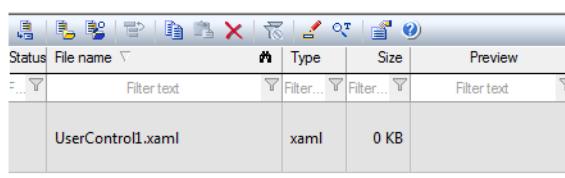
In the next step, how the DLL and XAML file are incorporated into zenon is explained.

## STEPS IN ZENON

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



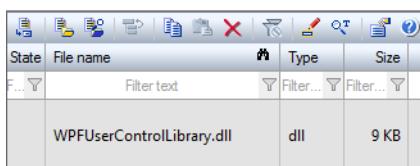
4. Select the XAML file at the save location (**UserControl1.xaml** from the **WPFUserControlNoCodeBehind** project) and add this:



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

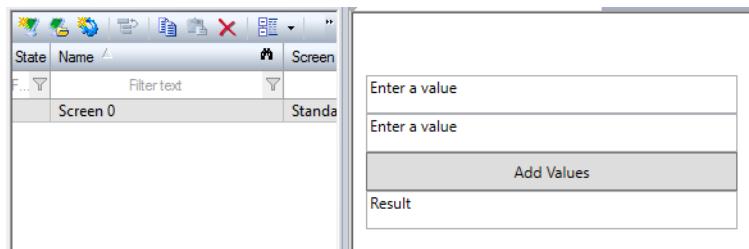
To do this:

- 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.



## Informations

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.

## 👉 Conseil

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 115) 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

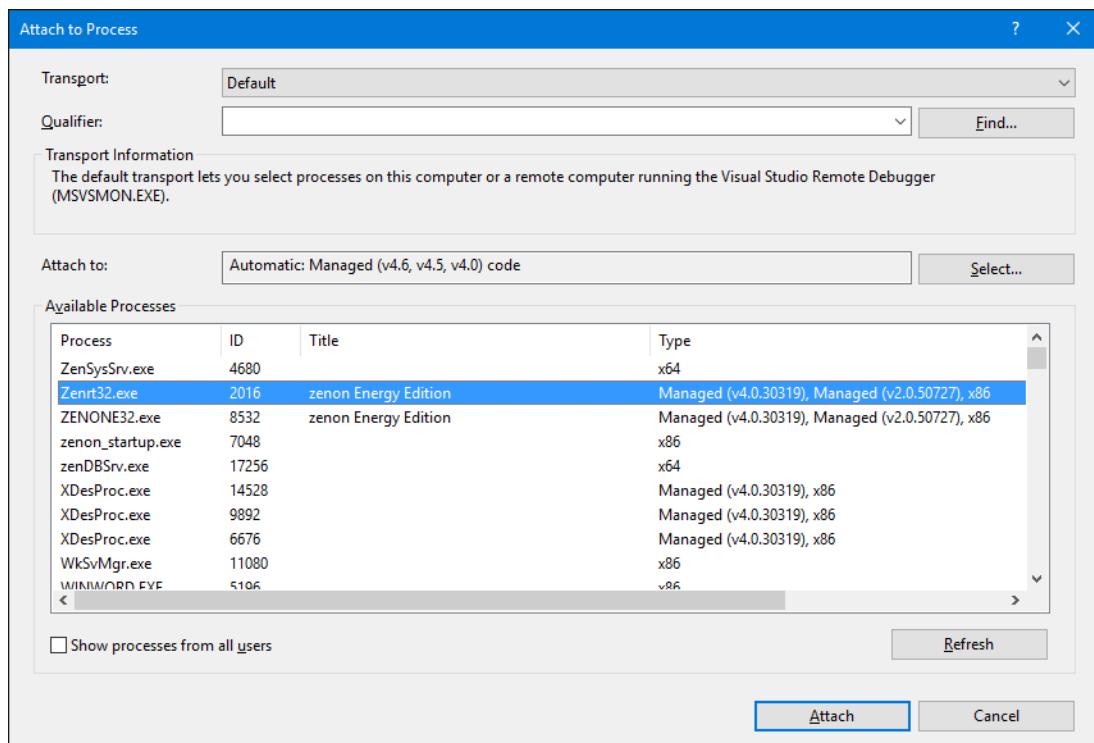
```

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     }

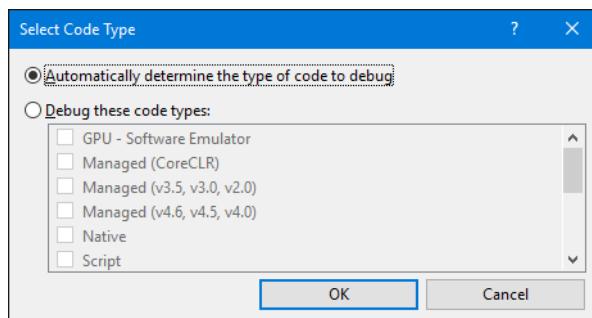
```

3. In Visual Studio, under **Debug**, select the **Attach to Process** menu item.

4. Select the zenon Runtime process

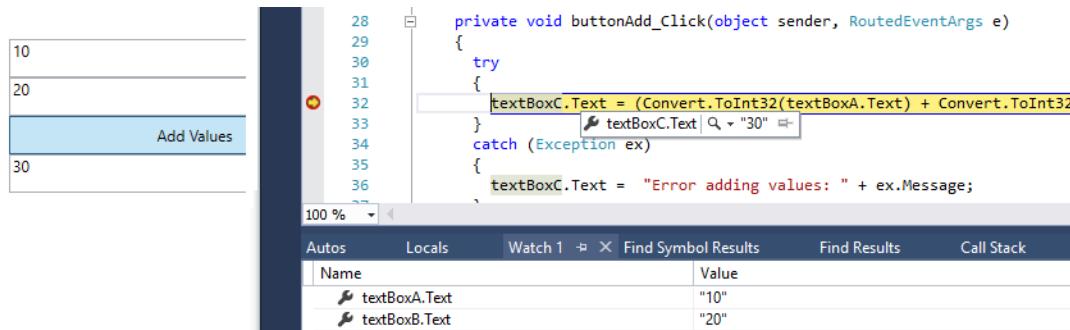


5. Under **Attach to**, select either **Automatic** or the corresponding .NET framework version



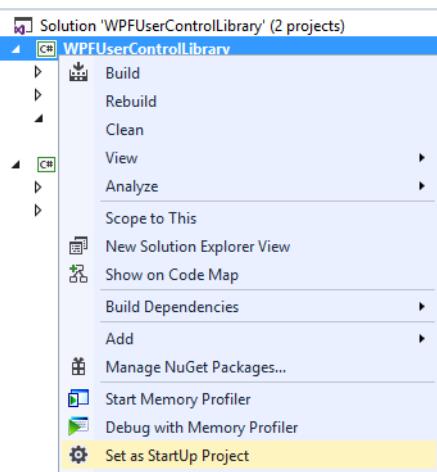
6. Click on **Attach**.

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



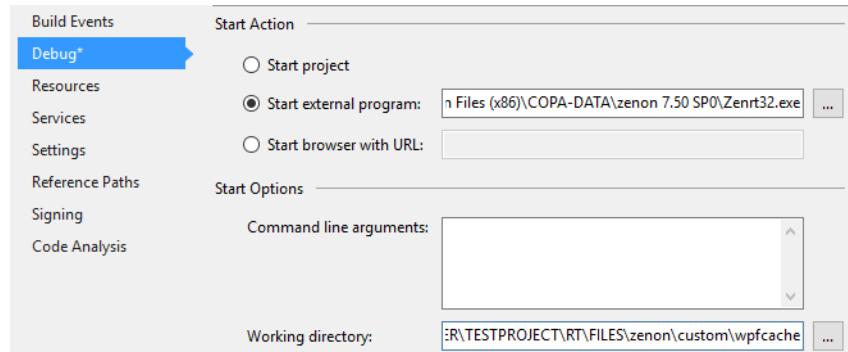
## DEBUG USING START EXTERNAL PROGRAM

- 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.

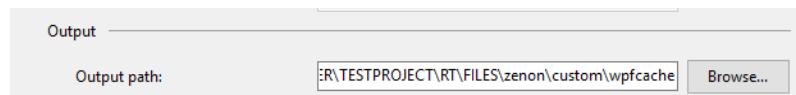


- In the project properties of the Visual Studio project, select under **Debug**, for **Start action:** **Start external program**
- For **Start external program**, select the path of the zenon Runtime application.
- Under Working Directory, select the **\wpfcache** folder of the Runtime files  
(...\PROJECTNAME\RT\FILES\zenon\custom\wpfcache)

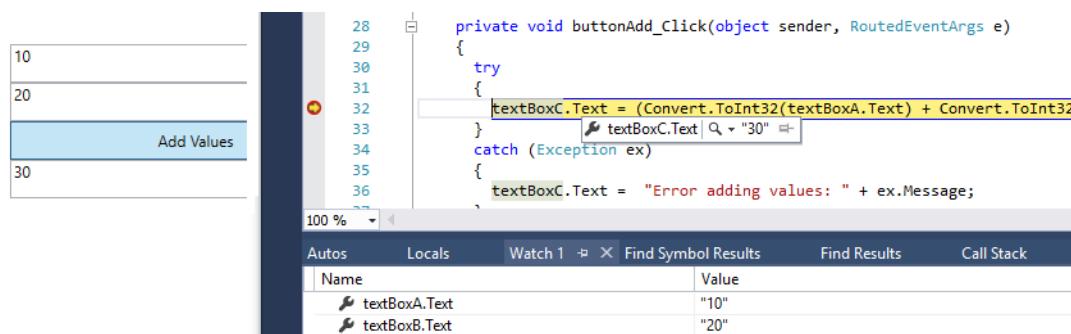
**Hint:** In the selected project in the zenon Editor, press the keyboard combination **CTRL+ALT+R** in order to jump directly to the root directory of the Runtime files.



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



## 💡 Informations

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).

## 👍 Conseil

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 **UserControl1.xaml** file

```
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)
{}
}
```

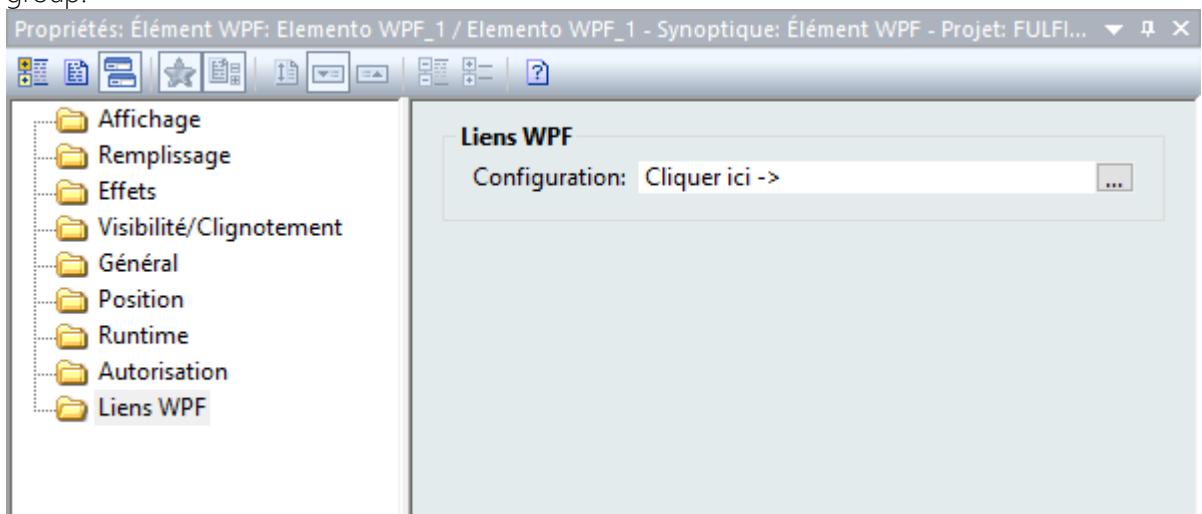
Then build the solution.

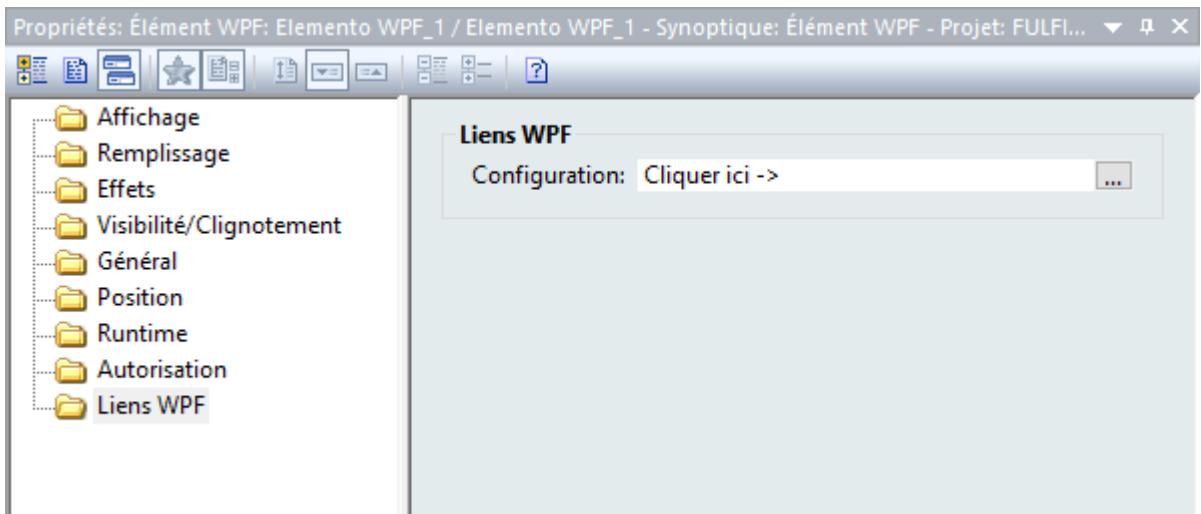
### Informations

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

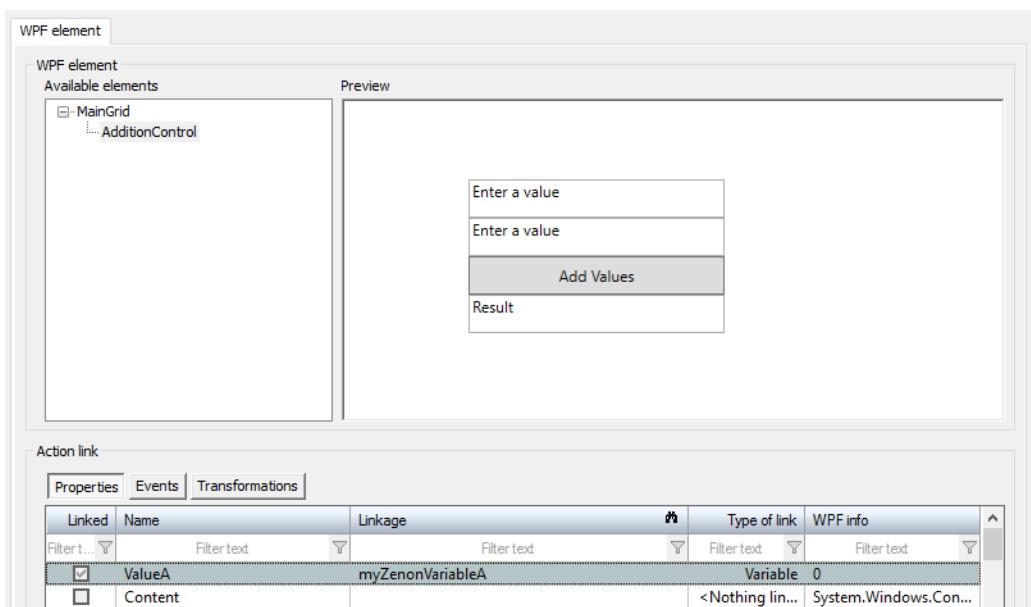
1. Update the WPF user control (DLL) in the zenon Editor.
2. Proceed as described in the creation of a simple WPF user controls with code behind function (à la page 24) chapter.
3. Create a numeric variable in zenon. Link the variable to a dynamic text element.  
You place the dynamic text element in the screen next to the WPF element with your user control.
4. Open the screen that contains the WPF element.
5. Go to **Configuration** in the properties of the WPF element in the **Liens WPF** property group.





6.

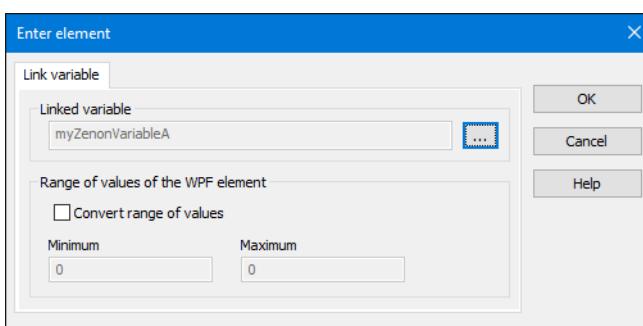
Expand the node in the tree view at the top left and select **AdditionControl**



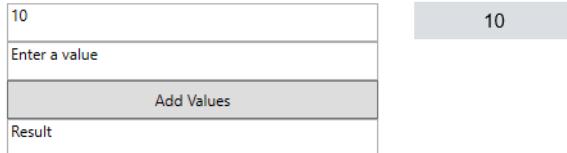
7. Select the line with *ValueA*(this is the name of the property that was created in the code beforehand) and select, for **Type of link: Variable**.

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

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



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



11. If the value is changed in user control, the value automatically changes in zenon and vice versa.
12. 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.

### Informations

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 Remplacement de données via VSTA

Les données peuvent également être échangées entre zenon et les contrôles utilisateur WPF à l'aide de VSTA.

Les méthodes d'éléments API suivantes :

- ▶ `get_WPFPProperty` (lecture des valeurs)
- ▶ `set_WPFPProperty` (écriture des valeurs)

sont utilisées à cette fin.

L'exemple utilisé ici repose sur l'exemple utilisé au chapitre Échange de données à l'aide des propriétés de dépendance (à la page 34).

## CRÉATION D'UNE MACRO VSTA POUR L'ÉCHANGE DE DONNÉES ENTRE ZENON ET LE CONTRÔLE UTILISATEUR WPF

- Créez la macro VSTA suivante dans le module complémentaire du projet zenon.

```

/// <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");

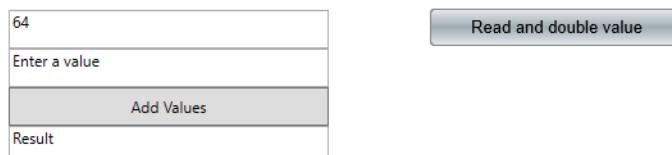
    //Read the current value from the WPF Element property
    double currentValue = Convert.ToDouble(myWPFElement.get_WPFProperty("AdditionControl", "ValueA"));

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

```

Dans ce cas de figure :

- ▶ "Screen" est le nom du synoptique de zenon dans lequel se trouve l'élément WPF
  - ▶ "WPF\_Element" est le nom de l'élément WPF contenant le contrôle utilisateur WPF
  - ▶ "AdditionControl" est le nom du contrôle utilisateur WPF lui-même (défini dans le fichier **UserControl1.xaml**)
  - ▶ "ValueA" est le nom de la propriété du contrôle utilisateur
- Créez et exécutez une fonction de macro VSTA et liez-la à un bouton, dans le synoptique, sur lequel se trouve également l'élément WPF.
  - Démarrez le Runtime pour tester les modifications



Lors de l'exécution de la macro, la valeur est lue par le contrôle, puis dupliquée et réécrite.

### 💡 Informations

Il n'est pas nécessaire que les propriétés du contrôle utilisateur utilisé pour cette méthode d'échange de données soient des propriétés de dépendance, comme indiqué dans cet exemple. Des propriétés "standard" peuvent également être utilisées, comme expliqué au chapitre Accès via une "liaison variable" de VSTA (à la page 40).

## 5.4 Zugriff auf das zenon (Runtime) Objektmodell aus einem WPF User Control

Für den Zugriff auf das zenon Objektmodell aus einem WPF User Control heraus, gibt es verschiedene Möglichkeiten. In den folgenden Kapiteln werden diese näher erläutert.

### ⚠ Attention

Lors de l'utilisation d'objets COM de zenon avec des contrôles utilisateur créés par l'utilisateur lui-même ou par des applications externes, ceux-ci doivent être activés avec la méthode **Marshal.ReleaseComObject**. L'activation avec la méthode **Marshal.FinalReleaseComObject** ne doit pas être utilisée, car elle entraîne un dysfonctionnement des modules Add-Ins de zenon.

### 5.4.1 Zugriff über VSTA "Variablen-Link"

Um Zugriff auf die zenon Runtime COM Schnittstelle per "Variablen-Link" zu bekommen gehen Sie wie folgt vor. Als Ausgangsbeispiel dient das im Kapitel Erstellung eines einfachen WPF User Controls mit Code-Behind Funktionalität (à la page 24).

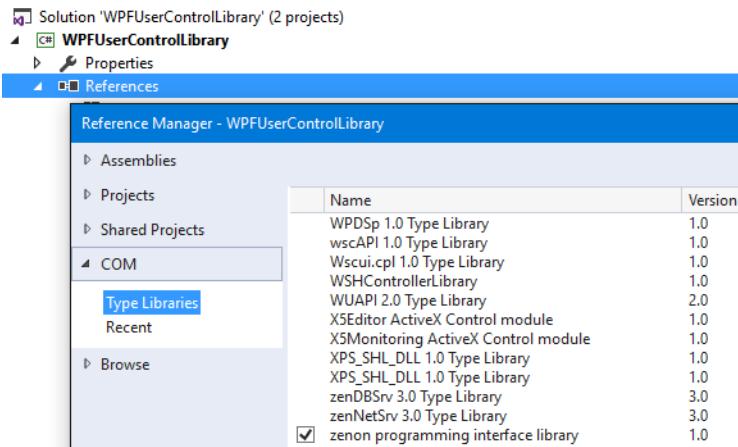
### 💡 Informations

Der folgende Code soll exemplarisch zeigen, wie der COM Zugriff auf die zenon Runtime realisiert werden kann und beschränkt sich dabei auf die Basisfunktionalität. Auf explizites Fehlerhandling, etc. wird verzichtet.

## NOTWENDIGE ANPASSUNGEN IM WPF USER CONTROL

Im WPF User Control Projekt (**WPFUserControlLibrary**) sind die folgenden Schritte notwendig.

Zunächst muss eine Referenz auf die zenon COM Schnittstelle eingebunden werden.



Danach muss in der Klasse `UserControl1` der folgende Code eingefügt werden:

```
//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();
        }
    }
}

```

Wobei auf die Properties `zenonVariableLink` (zur Initialisierung des COM Objekts) und `zenonReleaseTrigger` (zum Freigegeben des COM Objekts) später von VSTA aus (schreibend) zugegriffen wird.

Um den COM Zugriff schnell auf sehr einfachem Weg zu testen, kann im bestehenden Button Klick Event des User Controls die folgende Code Zeile eingefügt werden.

```
private void buttonAdd_Click(object sender, RoutedEventArgs e)
```

```
{
    if (zenonProject != null)
    {
        MessageBox.Show(zenonProject.Name);
    }
    return;
    ...
}
```

## 💡 Informations

In diesem Beispiel wird eine Variable vom Typ `zenOn.Project` verwendet. Es können natürlich auch andere Objekte sowie Events, etc. des zenon Objektmodels verwendet werden.

## NOTWENDIGE ANPASSUNGEN IM ZENON PROJEKT/VSTA CODE

Im VSTA Code sind die folgenden Schritte notwendig:

Erstellung eines VSTA-Makros für die Initialisierung

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

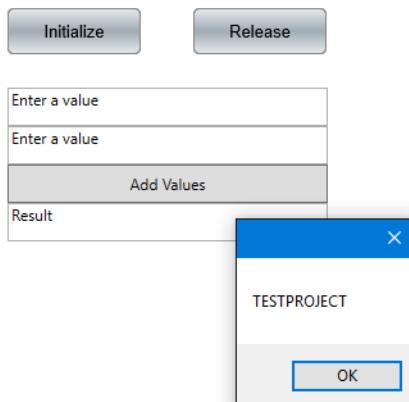
Erstellung eines VSTA-Makros für das Freigeben

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

Erstellen Sie zwei VSTA Makro ausführen Funktionen, die mit Buttons verknüpft werden, welche sich im selben Bild wie das WPF Element befinden.

Starten Sie nun die Runtime um die Funktionalität zu testen

- ▶ Führen Sie das Makro zum Initialisieren aus
- ▶ Betätigen Sie den Button im WPF User Control, es erscheint eine MessageBox mit dem Projektnamen des Projektes



- ▶ Führen Sie das Makro zum Freigeben aus

Um das User Control zu debuggen kann wie im Kapitel Debuggen des WPF User Controls zur Runtime (à la page 29) beschrieben vorgegangen werden.

### 👍 Conseil

Das Initialisieren und Freigeben des COM Objekts in diesem Beispiel erfolgt zur einfacheren Demonstration über VSTA Makro Funktionen. Abhängig vom Anwendungsfall, bzw. im praxisnahen Einsatz sind dafür Events in VSTA besser geeignet.

Beispielsweise kann der Code zum Initialisieren im `_open` Event des Bildes mit dem WPF Element ausgeführt werden und der Code zum Freigeben im `_close` Event.

Der hier beschriebene Mechanismus wird auch im Kapitel Anzeige von WPF-Elementen im zenon Web Client (à la page 112) verwendet.

### ⚠ Attention

Werden COM Objekte in WPF User Controls verwendet, müssen diese immer vor Zerstörung des WPF User Controls (vor dem Schließen des Bildes, vor dem Beenden der Runtime, vor dem Nachladen) explizit freigegeben werden.

## 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.

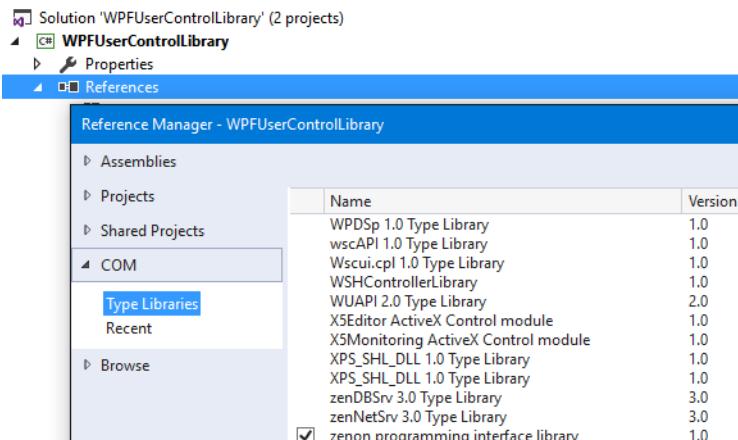
### 💡 Informations

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)
    {
    }
}
```

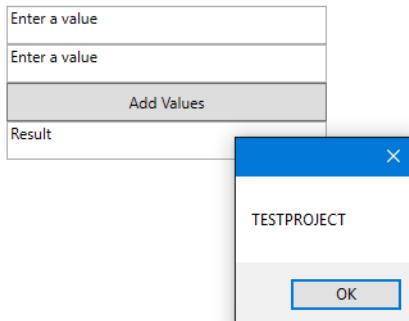
In order to test the COM access quickly very easily, it is possible to insert the following line of code in the existing button click event of the user control.

```
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).

### Informations

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).

### Informations

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 40) must be used.

## 6 Engineering in zenon

In order to be able to use WPF user controls in zenon, version 4.6.2 (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 Fichiers CDWPF (fichiers collectifs)

Un fichier CDWPF (avec le suffixe **\*.cdwfpf**) est un fichier ZIP renommé contenant les composants suivants :

- ▶ Fichier XAML (pour référencer l'ensemble de contrôle utilisateur)
- ▶ Fichier DLL (le contrôle utilisateur WPF lui-même, facultatif)
- ▶ Aperçu des éléments graphiques (pour l'aperçu, facultatif)

Règles d'utilisation des fichiers collectifs :

- ▶ Les fichiers (XAML, DLL, aperçu d'éléments graphiques) peuvent se trouver directement dans le fichier CDWPF ou dans un dossier associé.
- ▶ Le nom du fichier collectif doit correspondre aux noms définis au nom du fichier XAML.
- ▶ Un seul fichier XAML peut être présent à la fois.
- ▶ L'aperçu graphique doit être de petite taille, et ne pas mesurer plus de 64 pixels en hauteur.  
Nom du fichier d'aperçu : **preview.png** ou le nom du fichier XAML suivi du suffixe **.png**.
- ▶ Vous pouvez utiliser autant d'ensembles que vous le souhaitez. La distinction est établie sur la base de la version du fichier.
- ▶ Il n'est pas nécessaire que les fichiers collectifs contiennent un ensemble.
- ▶ Tous les sous-dossiers sont examinés et sont uniquement pris en compte avec les fichiers **\*.dll**, **\*.xaml** ou **\*.png**.
- ▶ Si un fichier collectif (**\*.cdwfpf**) est remplacé par une autre version du fichier, il est nécessaire d'adapter tous les fichiers CDWPF correspondants dans tous les projets.

## 6.2 WPF-Element anlegen

Um ein WPF-Element anzulegen:

1. Wählen Sie in der Symbolleiste der Elemente das Symbol für **WPF-Element** oder den entsprechenden Eintrag im Menü **Elemente**.
2. Wählen Sie im Hauptfenster den Startpunkt.
3. Ziehen Sie das Element mit der Maus auf.
4. Wählen Sie in den Eigenschaften in der Gruppe **Affichage** die Eigenschaft **Fichier XAML**.
5. Der Datei-Auswahldialog öffnet sich.
6. Wählen Sie die gewünschte Datei aus.  
Gültige sind Dateien vom Format:
  - ▶ \*.xaml: Extensible Application Markup Language
  - ▶ \*.cdwfpf: WPF Sammeldatei, zeigt auch Vorschaubild(Die Datei muss bereits im Projektmanager unter **Dateien/Grafiken** vorhanden sein oder im Dialog neu angelegt werden.)
7. Konfigurieren Sie die Verknüpfungen (à la page 50).

### Informations

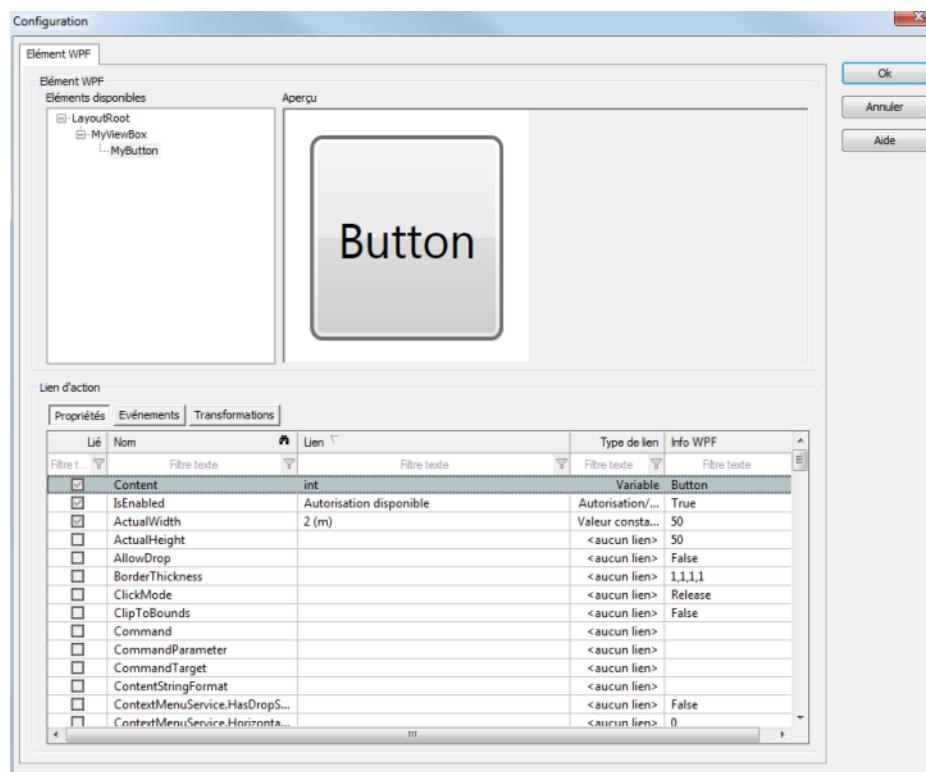
Sollen referenzierte Assemblies verwendet werden, beachten Sie die Hinweise im Kapitel Referenzierte Assemblies (à la page 8).

## 6.3 Konfiguration der Verknüpfung

Um ein WPF-Element zu konfigurieren

1. Wählen Sie in den Eigenschaften in der Gruppe **Liens WPF** die Eigenschaft **Configuration**.
2. Der Dialog mit drei Registerkarten öffnet sich mit einer Vorschau der XAML-Datei sowie der in der Datei vorhandenen Elemente.

### DIALOG KONFIGURATION



Parameter	Beschreibung
<b>Vorhandene Elemente</b>	<p>Zeigt in einer Baumstruktur die benannten Elemente der Datei an. Das gewählte Element kann mit Prozessdaten verknüpft werden.</p> <p>Die Zuordnung von <b>WPF</b> zur Prozessdaten basiert auf den Elementnamen. Daher werden Elemente nur angezeigt, wenn sie und die dazugehörigen übergeordneten Elemente einen Namen besitzen. Zuordnungen werden in den Registerkarten <b>Eigenschaften</b>, <b>Ereignisse</b> und <b>Transformationen</b></p>

Parameter	Beschreibung
	<p>konfiguriert und angezeigt.</p> <p><b>Tipp:</b> Werden die entsprechenden Elemente nicht angezeigt, überprüfen Sie, ob die nicht angezeigten Elemente in der XAML-Datei mit einem Namen versehen sind (z.B.: &lt;Grid Name="GridName"&gt;).</p>
<b>Vorschau</b>	Das ausgewählte Element wird in der Vorschau blinkend dargestellt.
<b>Eigenschaften</b> (à la page 52)	Konfiguration und Darstellung der Eigenschaften (Variablen, Bedienberechtigungen, Verriegelungen, verknüpfte Werte).
<b>Ereignisse</b> (à la page 59)	Konfiguration und Darstellung der Ereignisse (Funktionen).
<b>Transformationen</b> (à la page 60)	Konfiguration und Darstellung der Transformationen.
<b>Name</b>	Name der Eigenschaft.
<b>Verknüpfung</b>	Auswahl der Verknüpfung.
<b>Verknüpfungsart</b>	Art der Verknüpfung (Variable, Berechtigung, Funktion)
<b>WPF-Info</b>	Zeigt für Eigenschaften den aktuellen Wert im WPF-Content an. Für den Anwender ist direkt ersichtlich, um welche Art von Eigenschaft es sich handelt (Boolean, String, etc.).
<b>Verknüpft</b>	<p>Zeigt an, ob eine Eigenschaft aktuell verwendet wird.</p> <p>Falls nicht in der Ansicht enthalten, kann der Spalteneintrag über <b>Kontextmenü-&gt;Spaltenauswahl</b> ausgewählt werden.</p>

## 💡Informations

Im Konfigurationsdialog können nur logische Objekt angezeigt werden. Visuelle Objekte werden nicht angezeigt. Hintergrundinformationen dazu und wie visuelle Objekte dennoch dynamisiert werden können, lesen Sie im Abschnitt Zuordnung zenon Objekt zu WPF Content.

## VERKNÜPFUNGEN BEARBEITEN

Alle konfigurierten Verknüpfungen können über die Eigenschaften des Elements bearbeitet werden. Klicken Sie dazu auf das Element und öffnen Sie die Eigenschaftengruppe **Liens WPF**. Hier können Sie Verknüpfungen weiter konfigurieren, ohne den Dialog öffnen zu müssen.

Limitations :

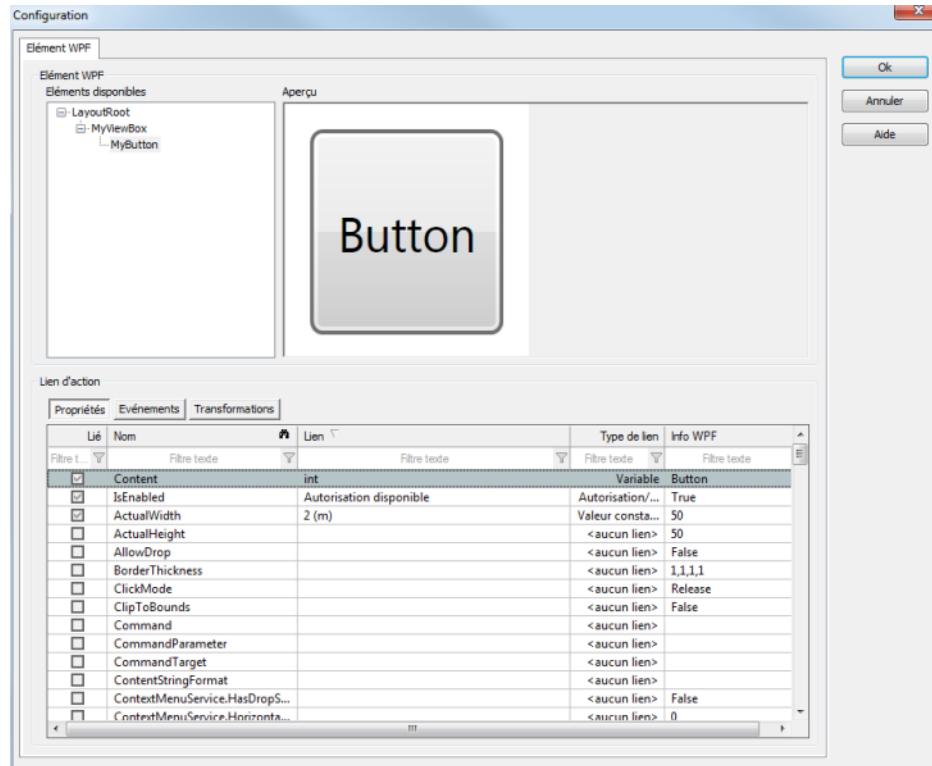
- ▶ Le type de liaison ne peut pas être modifié ici.
- ▶ Les nouveaux liens peuvent uniquement être créés via la boîte de dialogue de configuration.
- ▶ Insertion d'un élément WPF dans un symbole : Les liens WPF ne peuvent pas être exportés.

### 6.3.1 Properties

The properties enable the linking of:

- ▶ Variables (à la page 54)
- ▶ Values (à la page 56)

- ▶ Authorizations and interlockings (à la page 57)



Parameter	Description
Name	Name of the property.
Lien	Linked variable, authorization or linked value.  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.  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

### Informations

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

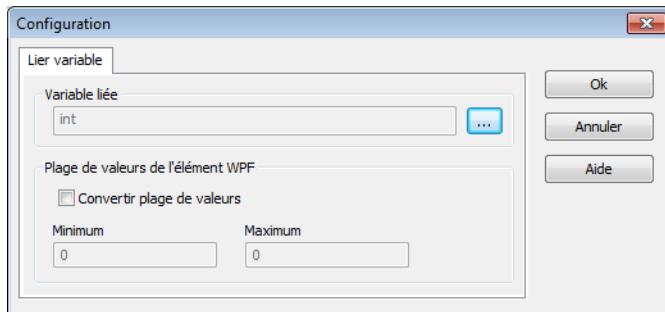
These values must be displayed via the linked zenon variable.

#### 6.3.1.1 Variable verknüpfen

Um eine Variable mit einer WPF-Eigenschaften zu verknüpfen:

1. markieren Sie die Zeile mit der Eigenschaft, die verknüpft werden soll
2. klicken Sie in die Zelle **Verknüpfungsart**
3. wählen Sie aus der Combobox **Variable**
4. klicken Sie in die Zelle **Verknüpfung**
5. der Dialog zur Konfiguration der Variablen öffnet sich

Dieser Dialog gilt auch für Auswahl der Variablen bei Transformationen (à la page 60). Die Konfiguration ermöglicht auch Umrechnung von zenon in WPF-Einheiten.



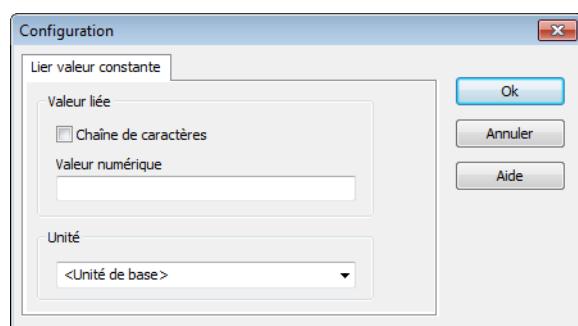
Parameters	Description
<b>Linked variable</b>	Selection of the variable to be linked. A click on the ... button opens the selection dialog.
<b>Range of values of the WPF element</b>	Data to convert variable values into WPF values.
<b>Convert range of values</b>	<p>Active: WPF unit conversion is switched 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 Werte verknüpfen

Verknüpfte Werte können entweder ein **String** oder ein numerischer Wert vom Typ **Double** sein. Beim Aufschalten des Bildes wird nach dem Laden des WPF-Contents der gewählte Wert im WPF-Content abgesetzt.

Um einen Wert mit einer WPF-Eigenschaft zu verknüpfen:

1. markieren Sie die Zeile mit der Eigenschaft, die verknüpft werden soll
2. klicken Sie in die Zelle **Verknüpfungsart**
3. wählen Sie aus der Combobox **Werteverknüpfungen**
4. klicken Sie in die Zelle **Verknüpfung**
5. der Dialog zur Konfiguration der Werteverknüpfung öffnet sich



Parameter	Beschreibung
<b>Verknüpfter Wert:</b>	Eingabe eines numerischen Werts oder Stringwerts.
<b>String verwenden</b>	<p>Aktiv: Ein String-Wert wird statt eines numerischen Werts verwendet.</p> <p>Stringwerte sind sprachumschaltbar. In der Runtime wird der Text beim Aufschalten des Bildes übersetzt und im WPF-Content abgesetzt. Wird während das Bild geöffnet ist eine Sprachumschaltung durchgeführt, wird der Stringwert neu übersetzt und abgesetzt.</p>
<b>Stringwert/Numerischer Wert</b>	Abhängig von der Auswahl der Eigenschaft <b>String verwenden</b> wird in dieses Feld ein numerischer Wert oder ein Stringwert eingegeben. Bei numerischen Werten kann zusätzlich eine Einheit Maßeinheit ausgewählt werden.
<b>Einheit:</b>	Auswahl einer Maßeinheit aus der Dropdownliste. Diese muss zuvor in der Einheitenumschaltung definiert worden sein.

Parameter	Beschreibung
	Die Maßeinheit wird dem numerischen Wert zugeordnet. Wird in der Runtime eine Einheitenumschaltung durchgeführt, wird der Wert umgerechnet auf die neue Maßeinheit und zum WPF-Content abgesetzt.

### FERMER LA BOÎTE DE DIALOGUE

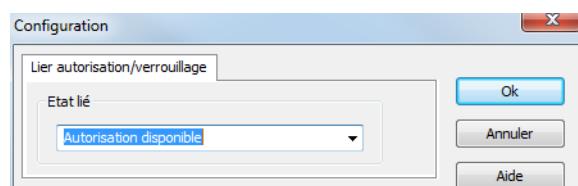
Options	Description
OK	Applique les paramètres et ferme la boîte de dialogue.
Annuler	Annule toutes les modifications et ferme la boîte de dialogue.
Aide	Ouvre l'aide en ligne.

### 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.

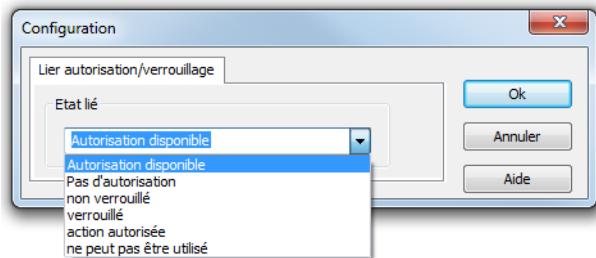
To link a user authorization or interlocking to a WPF property:

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

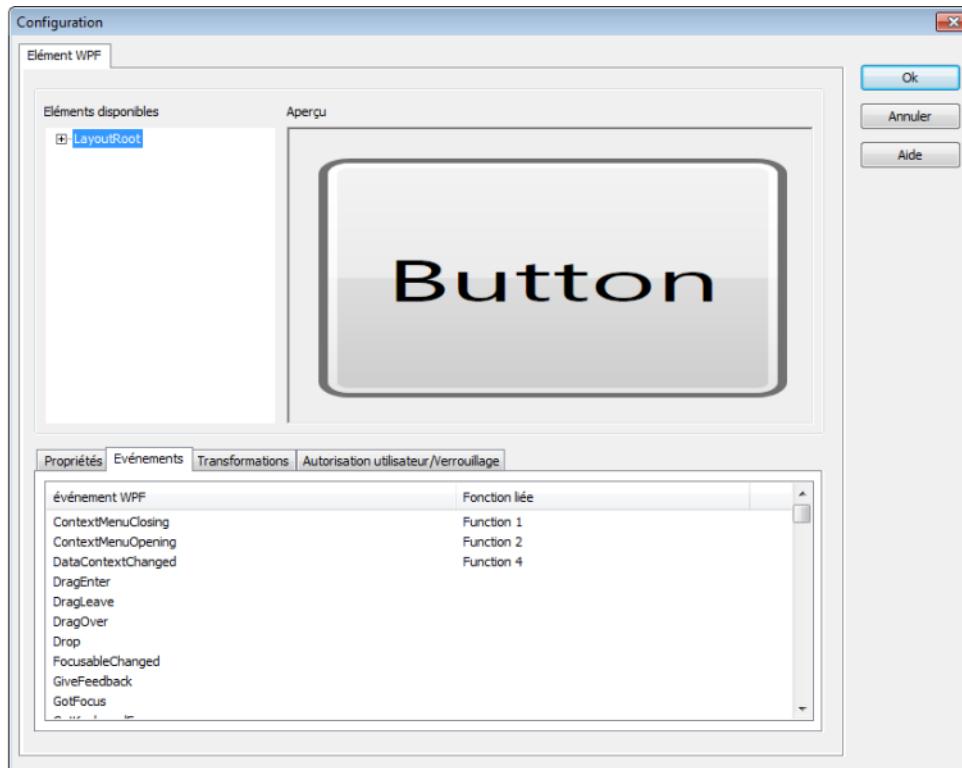
Parameter	Description
	control from the drop down list. For example, visibility and operability of a WPF button can depend on a user's status.



Authorization	Description
<i>Operating authorization available</i>	If the user has sufficient rights to operate the <b>WPF element</b> , a value of 1 is written to the property.
<i>Operating 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.

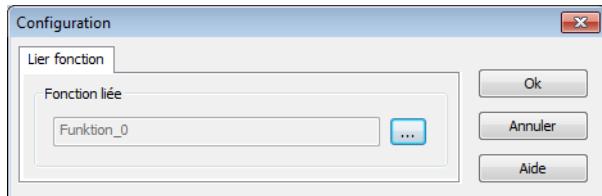


Parameter	Description
<b>Name</b>	Name of the property.
<b>Linkage</b>	Linked function. Clicking in the cell opens the configuration dialog.
<b>Type of link</b>	Selection of linking. Clicking in the cell opens the selection dialog.
<b>WPF info</b>	Shows the current value for properties in WPF content. 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. 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

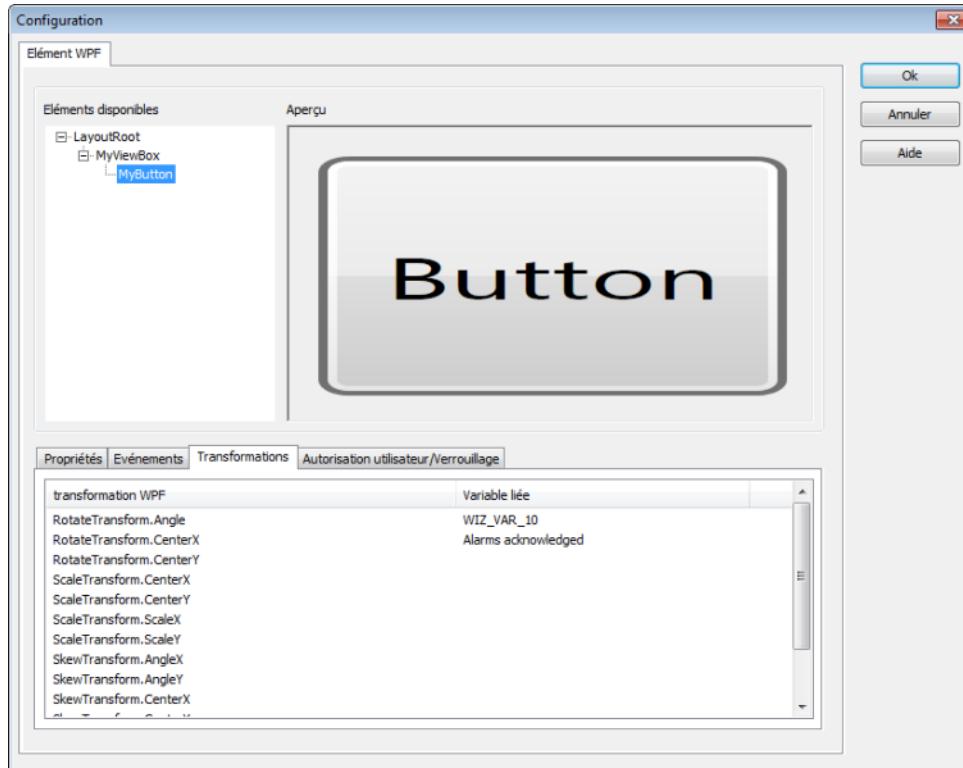


Parameter	Description
<b>Linked function</b>	Selection of the function to be linked. Clicking on the ... button opens the dialog for Function selection.
<b>OK</b>	Accepts selection and closes dialog.
<b>Cancel</b>	Discards changes and closes dialog.
Help	Opens online help.

### 6.3.3 Transformation

L'**élément WPF** ne prend pas en charge la rotation. Par exemple, si l'**élément WPF** se trouve dans un symbole et ce symbole subit une rotation, l'**élément WPF** ne pivotera pas avec le symbole. Par conséquent, il existe un autre mécanisme de **Transformation** dans WPF permettant de faire pivoter les éléments ou de les transformer autrement. Ces transformations sont configurées dans l'onglet **Transformation**.

**Attention :** si le contenu se trouve hors de la zone de l'**élément WPF**, cette partie du contenu est perdue ou n'est pas affichée.



Paramètre	Description
<b>Nom</b>	Nom de la propriété.
<b>Lien</b>	<p>Sélection des variables liées.</p> <p>Les transformations sont affichées en langage XAML sous forme d'objets de transformation possédant des propriétés propres. Si un élément est compatible avec une transformation, les propriétés possibles de l'objet de transformation sont affichées dans la liste (pour plus d'informations à ce sujet : Intégration d'un bouton sous forme d'élément WPF XAML dans zenon (à la page 120)</p> <p>Par exemple, si la variable liée est définie à une valeur de 10, cette valeur est écrite sous forme de cible WPF, et l'élément WPF est pivoté de 10°.</p>
<b>Type de lien</b>	Sélection du type de lien de transformation.
<b>Informations WPS</b>	Affiche la valeur actuelle des propriétés du contenu WPF. L'utilisateur peut directement visualiser le type de propriété dont il s'agit (valeur booléenne, chaîne, etc.).
<b>Liée</b>	Indique si une propriété est actuellement utilisée.

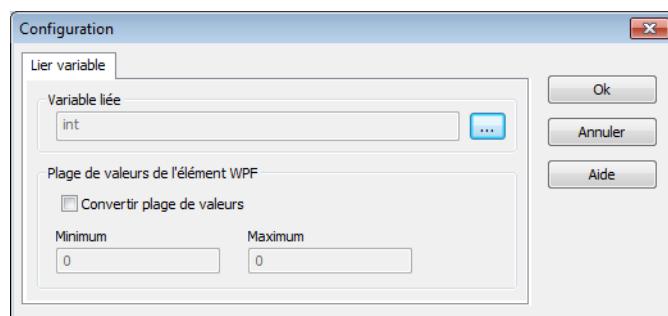
Paramètre	Description
	Si non contenue dans la vue, peut être sélectionnée via <b>Menu contextuel&gt;en sélectionnant Colonne.</b>

## LIEZ LES TRANSFORMATIONS

Pour lier une transformation avec une propriété WPF :

1. Sélectionnez la ligne comportant la propriété que vous souhaitez lier.
2. Cliquez sur la cellule **Link type (Type de lien)**.
3. Sélectionnez **Transformation** dans la liste déroulante.
4. Cliquez sur la cellule **Lien**.
5. La boîte de dialogue de configuration des variables s'affiche à l'écran.

La configuration permet également d'effectuer des conversions en unités WPF depuis zenon.



Parameters	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</li> </ul>

Parameters	Description
	to the variables ▶ <b>Adjust value</b> activated ▶ a WPF value range of 0 to 360 is configured  Now the rotary knob can be turned at a value of 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 Validité des fichiers XAML

La validité des fichiers XAML est soumise à certaines exigences :

- ▶ Espace de nommage correct
- ▶ Absence de référence de classe
- ▶ Capacité d'extension

### ESPACE DE NOMMAGE CORRECT

L'**élément WPF** peut uniquement afficher un contenu WPF, c'est-à-dire :

seuls les fichiers XAML comportant un espace de nommage WPF de longueur correcte peuvent être affichés par l'**élément WPF**. Les fichiers utilisant un espace de nommage Silverlight ne peuvent être ni chargés, ni affichés. Toutefois, dans la plupart des cas, il suffit de convertir l'espace de nommage Silverlight en espace de nommage WPF.

Espaces de nommage WPF :

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

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

## ABSENCE D'UTILISATION DE RÉFÉRENCES DE CLASSE

Les fichiers XAML pouvant être chargés de façon dynamique, il est impossible d'utiliser des fichiers XAML contenant des références à de classes (clé "class" dans l'en-tête). Les fonctions programmées dans des fichiers C# créés indépendamment ne peuvent pas être utilisées.

Pour utiliser des contrôles WPF avec une fonction code-behind, suivre la procédure décrite au chapitre Création d'un contrôle utilisateur WPF simple avec fonction code-behind (à la page 24).

## CAPACITÉ D'EXTENSION

Si le contenu d'un **élément WPF** est ajusté à la taille de l'**élément WPF**, les commandes de l'**élément WPF** sont entrelacées dans une commande offrant cette fonctionnalité, à l'image d'une **Viewbox**, par exemple. En outre, il convient de veiller que les éléments **hauteur** et **largeur** soient configurés de manière **automatique**.

## VÉRIFICATION DE LA VALIDITÉ D'UN FICHIER XAML

Pour s'assurer qu'un fichier XAML possède le format adéquat :

- ▶ Ouvrez le fichier XAML dans Internet Explorer.
  - ▶ S'il peut être ouvert sans modules d'extension supplémentaires (Java ou autres), on peut supposer avec certitude que ce fichier peut être chargé et affiché dans zenon.
  - ▶ Si un problème survient durant le chargement, celui-ci est affiché dans Internet Explorer et les lignes à l'origine du problème sont indiquées de manière claire.

La capacité d'extension peut également être testée de cette manière : Si le fichier a été correctement créé, le contenu sera automatiquement ajusté à la taille de la fenêtre Internet Explorer.

## MESSAGE D'ERREUR

Si un fichier non valide est utilisé dans zenon, un message d'erreur est affiché dans la fenêtre de sortie lors du chargement du fichier dans l'élément WPF.

Par exemple :

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

*L'attribut "Classe" ne peut être trouvé dans l'espace de nommage XML  
"http://schemas.microsoft.com/winfx/2006/xaml". Ligne 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 66)
- ▶ Vertical bar graph (à la page 66)
- ▶ BACnet WPF Control
- ▶ Comtrade Viewer (à la page 68)
- ▶ Energy class diagram (à la page 81)
- ▶ Progress bar (à la page 67)
- ▶ Pareto diagram (à la page 82)
- ▶ Sankey Diagram (à la page 89)
- ▶ Round display (à la page 85)
- ▶ Temperature control (à la page 90)
- ▶ Universal slider (à la page 92)
- ▶ Waterfall chart (à la page 94)

## 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 Horloge analogique - AnalogClockControl

Propriété	Fonction	Valeur
<b>ElementStyle</b>	Forme/type d'élément.	<p><i>Enum :</i></p> <ul style="list-style-type: none"> <li>▶ SmallNumbers</li> <li>▶ BigNumbers</li> <li>▶ Non</li> </ul>
<b>ElementBackgroundBrush</b>	Couleur de l'arrière-plan de l'élément.	<i>Pinceau</i>
<b>ElementGlasReflection</b>	Active l'effet vitré sur l'élément.	<i>Visibilité</i>
<b>Offset</b>	Valeur en heures (h) affichant le décalage de temps par rapport à l'horloge système.	<i>Int16</i>
<b>OriginText</b>	Texte affiché sur l'horloge (par exemple, emplacement).	<i>String</i>

### 6.5.2 Bargraphe vertical - VerticalBargraphControl

Propriété	Fonction	Valeur
<b>CurrentValue</b>	Valeur actuelle à afficher.	<i>Double</i>
<b>MinValue</b>	Valeur minimum de l'échelle.	<i>Double</i>
<b>MaxValue</b>	Valeur maximum de l'échelle.	<i>Double</i>
<b>MajorTicksCount</b>	Nombre de graduations principales sur l'échelle.	<i>Integer</i>
<b>MinorTicksCount</b>	Nombre de graduations secondaires sur l'échelle.	<i>Integer</i>
<b>MajorTickColor</b>	Couleur des graduations principales sur l'échelle.	<i>Couleur</i>
<b>MinorTickColor</b>	Couleur des graduations secondaires sur l'échelle.	<i>Couleur</i>
<b>ElementBorderBrush</b>	Couleur de la bordure de l'élément.	<i>Pinceau</i>
<b>ElementBackgroundBrush</b>	Couleur de l'arrière-plan de l'élément.	<i>Pinceau</i>
<b>ElementGlasReflection</b>	Active l'effet vitré sur l'élément.	<i>Visibilité</i>

Propriété	Fonction	Valeur
<b>ElementFontFamily</b>	Police de caractères de l'élément.	<i>Police</i>
<b>ScaleFontSize</b>	Taille de la police de caractères de l'élément.	<i>Double</i>
<b>ScaleFontColor</b>	Couleur de la police de caractères de l'échelle.	<i>Couleur</i>
<b>IndicatorBrush</b>	Couleur de remplissage du graphique à barres.	<i>Pinceau</i>
<b>BargraphSeparation</b>	Nombre de divisions du bargraphe.	<i>Integer</i>
<b>BargraphSeparationColor</b>	Couleur des divisions de l'échelle.	<i>Couleur</i>

### 6.5.3 Fortschrittsanzeige - ProgressBarControl

Property	Funktion	Wert
<b>CurrentValue</b>	Aktueller Wert, der angezeigt werden soll.	<i>Double</i>
<b>MinValue</b>	Minimalwert des Wertebereichs.	<i>Double</i>
<b>MaxValue</b>	Maximalwert des Wertebereichs.	<i>Double</i>
<b>ProgressbarDivisionCount</b>	Anzahl der Unterteilungen der (Boxed) Fortschrittsanzeige.	<i>Integer</i>
<b>VisibilityText</b>	Sichtbarkeit der Wertanzeige.	<i>Boolean</i>
<b>TextSize</b>	Schriftgröße der Wertanzeige.	<i>Double</i>
<b>TextColor</b>	Farbe der Wertanzeige.	<i>Color</i>
<b>ProgressBarBoxedColor</b>	Farbe der Rahmen der (Boxed) Fortschrittsanzeige.	<i>Color</i>
<b>ProgressBarMarginDistance</b>	Abstand der Progressbar-Box vom Elementrand (links, oben, rechts, unten).	<i>Double</i>
<b>ProgressBarInactiveBrush</b>	Indikatorfarbe nicht aktiv.	<i>Brush</i>
<b>ProgressBarActiveBrush</b>	Indikatorfarbe aktiv.	<i>Brush</i>
<b>ProgressBarPadding</b>	Abstand der Fortschrittsanzeige von der Progressbar-Box (links, oben, rechts, unten).	<i>Double</i>
<b>ElementBorderBrush</b>	Farbe des Elementrahmens.	<i>Brush</i>
<b>ElementBackgroundBrush</b>	Farbe des Elementhintergrunds.	<i>Brush</i>

Property	Funktion	Wert
h		

## 6.5.4 COMTRADE-Viewer

The **COMTRADE-Viewer** WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.

### Informations de licence

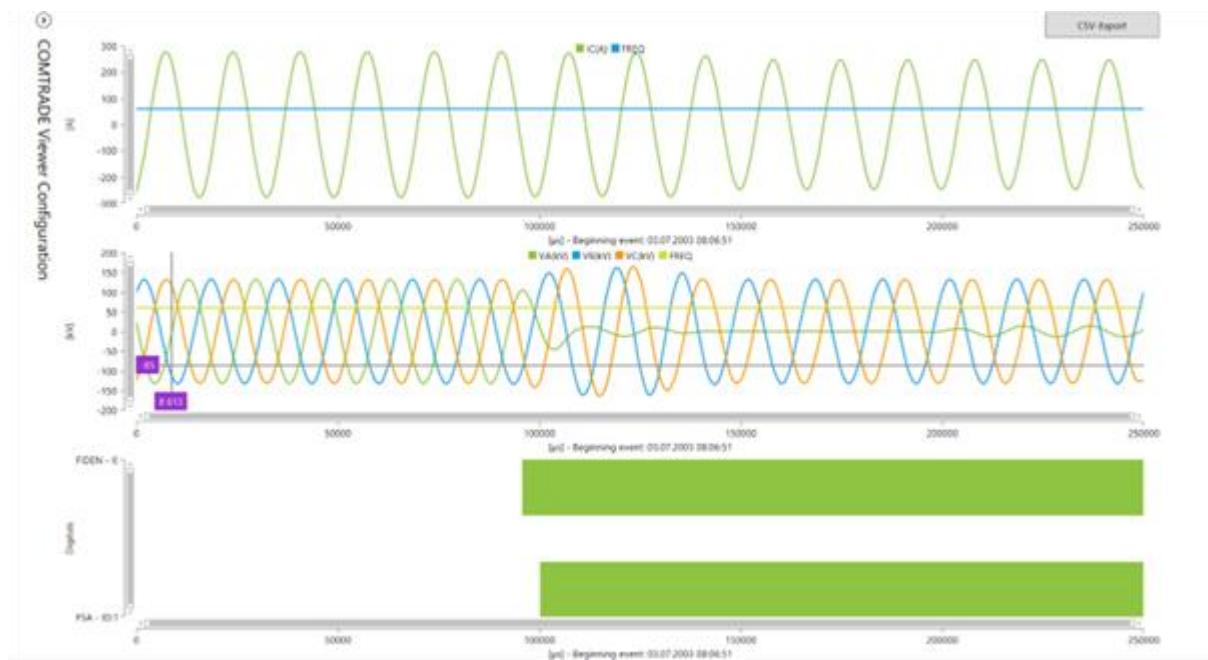
The **COMTRADE-Viewer** can only be configured in the zenon Editor with a valid Energy Edition license. If there is no valid license, the WPF is displayed as grayed out in the Editor. A valid Energy Edition license is also required for display in the Runtime.

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

### Informations

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.

## 👉 Conseil

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

## 💡 Informations

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 Configurable control properties - color display

### ENGINEERING IN THE EDITOR

The element with the name **COMTRADE.CDWPF** can be configured and placed in each zenon screen type.

The project configuration of **Largeur [pixels]** and **Hauteur [pixels]** of the element depend on the proportions. This prevents the **COMTRADE-Viewer** being displayed as distorted in Runtime.

**Note:** When configuring the project, ensure that there is sufficient size to guarantee a clear overview.

### GRAPHICAL AMENDMENTS

You configure the graphic design in the properties of the WPF element.

You can find further information in the configuration of the linking (à la page 50) chapter in this manual.

Possible color values:

- ▶ Hexadecimal color values in the following formats can be used:  
#RRGGBB and #AARRGGBB  
Transparency values are permitted.
- ▶ **Example color values:**  
#000000 = black  
#FFFFFF = white  
#FF0000 = red
- ▶ Color values by name  
Reference: <https://msdn.microsoft.com/en-us/library/system.drawing.color.aspx>  
<https://msdn.microsoft.com/en-us/library/system.drawing.color.aspx>

#### Conseil

The properties for the **COMTRADE-Viewer** WPF element have a "z" as a starting color. Use name filtering for a clear display when configuring the linking.

### CONFIGURATION PAGE

Text and background color of the configuration page.

#### Analog Channels

Parameter	Description	Value
<b>zConfiguratinPageTextColor</b>	Text color of the configuration page	<i>String</i>

Parameter	Description	Value
<b>zConfigurationPageBackgroundColor</b>	Background color of the configuration page	<i>String</i>

## BUTTONS

Text and background color of the button.

Open...

Parameter	Description	Value
<b>zButtonTextColor</b>	Text color of the button	<i>String</i>
<b>zButtonBackgroundColor</b>	Background color of the button	<i>String</i>

## CHART

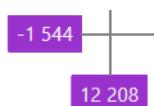
Text color of the axis labeling or key and background color.

50000                    60000                    70000

Parameter	Description	Value
<b>zChartTextColor</b>	Text color of the axis labeling.	<i>String</i>
<b>zChartBackgroundColor</b>	Background color of the axis labeling	<i>String</i>

## LABEL

Text and background color of the display of a selected cursor position.



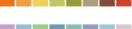
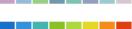
Parameter	Description	Value
<b>zChartLabelTextColor</b>	Text color of the value display	<i>String</i>
<b>zChartLabelBackgroundColor</b>	Background color of the value display	<i>String</i>

## CHART

Color palette of the graph view and the attendant keys.

Parameter	Description	Value
<b>zChartPalette</b>	<p>Color palette of the colors for graphs and keys.</p> <p>Referencing with color palette name (see overview).</p> <p>Par défaut : if no color palette is configured, the color palette of the computer's operating system is used.</p>	<i>String</i>

### POSSIBLE COLOR PALETTES - OVERVIEW

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

### 6.5.4.5 Konfigurierbare Control-Eigenschaften - Sprach- und Anzeigeeinstellungen

Über die folgenden Einstellungen können Sie definieren:

- ▶ ob zum Öffnen von Windows-Dateien der Standarddialog oder ein benutzerdefinierter Dialog verwendet werden soll.
- ▶ ob die Benutzeroberfläche in Polnisch dargestellt werden soll.
- ▶ ob der **CSV-Export** Button in der Runtime angezeigt werden soll.

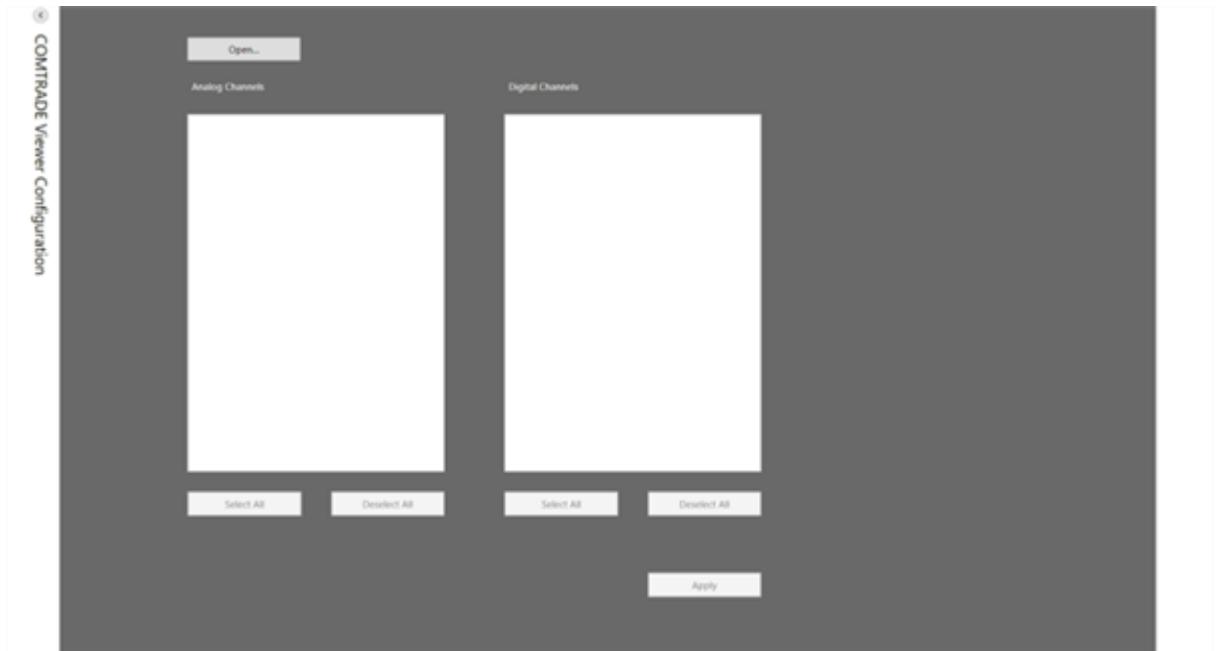
Parameter	Beschreibung
<b>zLimitFileOpenAccessTo</b>	Wenn der Parameter auf einen anderen Wert als leer gesetzt wird, öffnet sich ein benutzerdefinierter Dateiöffnungsdialog zum Öffnen von Windows

Parameter	Beschreibung
	<p>Dateien.</p> <p>Dieser Dialog zeigt alle <b>*.CFG</b> Dateien an, bei denen eine entsprechende <b>*.DAT</b> Datei existiert in dem Ordner an, der durch den Parameter adressiert ist.</p> <p>Default: <i>leer</i></p>
<b>zUiLanguage</b>	<p>Wenn der Parameter auf einen der folgenden Werte gesetzt wird, werden die Benutzeroberfläche des <b>COMTRADE-Viewer</b> und der benutzerdefinierte Dateiöffnungsdialog in Polnisch angezeigt:</p> <ul style="list-style-type: none"> <li>▶ <i>pl</i></li> <li>▶ <i>pL</i></li> <li>▶ <i>Pl</i></li> <li>▶ <i>PL</i></li> </ul> <p><b>Hinweis:</b> Beim Standarddialog zum Öffnen von Dateien wird nur der Titel auf Polnisch angezeigt. Die anderen Einträge beziehen sich auf die Windows Spracheinstellungen.</p> <p>Default: <i>en</i></p>
<b>zButtonCSVExportVisibility</b>	<p>Definiert die Anzeige des <b>CSV-Export</b> Buttons in der Runtime:</p> <ul style="list-style-type: none"> <li>▶ <i>True</i>: Button wird angezeigt</li> <li>▶ <i>False</i>: Button wird nicht angezeigt</li> </ul> <p>Default: <i>True</i></p>

#### 6.5.4.6 Runtime Ansicht - Konfigurationsseite

Wird in der Runtime ein Bild mit einem projektierten **COMTRADE-Viewer** WPF-Element aufgeschalten so ist die Anzeige der Konfigurationsseite jeweils leer.

**Hinweis:** Dies gilt auch, wenn in der zenon Runtime von einem anderen Bild zu diesem Bild mit der Bildumschaltfunktion gewechselt wird.



## COMTRADE VIEWER CONFIGURATION

Die seitlich vertikal angeordnete Umschaltung **COMTRADE Viewer Configuration** wechselt die Anzeige von der Konfiguration zur Graphen Ansicht und umgekehrt.

## DATEI AUSWÄHLEN

Der Button **Open...** öffnet den Dateiauswahldialog zur Auswahl einer Datei.

Für die Anzeige im Dateiauswahldialog findet eine Vorauswahl statt:

- ▶ Dabei werden Dateipaare von \*.cfg- und \*.dat-Dateien erkannt.  
**Hinweis:** Optionale Dateien vom Dateityp \*.hdr oder \*.inf werden nicht berücksichtigt.
- ▶ Angezeigt werden nur die entsprechenden \*.dat-Dateien.
- ▶ Mit Klick auf die gewünschte Datei und Klick auf den Button **OK**, werden alle zugehörigen Dateien (\*.dat, \*.cfg) geladen.
- ▶ Es kann jeweils nur eine Datei geladen werden.
- ▶ Nach Laden der Datei werden die Inhalte der Datei in den Spalten **Analog Channels** und **Digital Channels** angezeigt.  
Die Bezeichnungen und Einheiten der Elemente haben Ihren Ursprung in der **COMTRADE** Konfiguration und können nicht geändert werden.

## WEITERFÜHRENDE INFORMATIONEN ZUR VERARBEITUNG VON **\_.CFG-** UND **\_.DAT-DATEIEN**

Die Informationen der \*.cfg-Datei ermöglichen die Auswertung der \*.dat-Datei. Diese beinhaltet die Daten der diversen analogen und digitalen Messreihen von Strömen und Spannungen. Die Daten sind in einzelne Datensätze unterteilt und im Hex-Format dargestellt.

### **\*.cfg-Dateien**

- ▶ Der letzte Eintrag einer Datei dieses Datentyps ist ein Zeitmultiplikator. Dieser Eintrag wird beim Einlesen eines Störschreibs mit dem Zeitstempel eines jeden Eintrags aus der \*.dat-Datei multipliziert. Falls kein Zeitmultiplikator vorhanden sein sollte, wird intern ein Faktor mit dem Wert 1 angenommen. Die \*.cfg-Datei wird dabei nicht verändert.
- ▶ Für die Einträge der digitalen Messwerte gelten gewisse Normen. Beispiel für einen standardkonformen Eintrag eines digitalen Messwerts: 1,LOPHC,,0. Falls die Null am Ende des Eintrags jedoch nicht vorhanden sein sollte, ergänzt der **COMTRADE-Viewer** diese intern. Die \*.cfg-Datei wird dabei nicht verändert.

### **\*.dat-Dateien**

- ▶ Der **COMTRADE-Viewer** ist in der Lage Dateien dieses Datentyps einzulesen, die mit dem Index 0 oder >1 beginnen. Dabei wird stets geprüft, ob diese Datensätze fortlaufend in diskreten Schritten von 1 nummeriert sind. Bei nicht korrekt nummerierten Datensätzen kann die Datei nicht eingelesen werden.

## ANALOG CHANNELS

Parameter	Beschreibung
[Liste der verfügbaren Kanäle]	Auswahl der zu visualisierenden Elemente.  Mehrfachauswahl mit Klick auf den gewünschten Eintrag in der Liste. Ausgewählte Elemente werden farbig hinterlegt dargestellt. Ein nochmaliger Mausklick hebt die Auswahl des Eintrags auf.
<b>Select All</b>	Sélectionne tous les éléments dans la liste.
<b>Deselect All</b>	Désactive la sélection d'éléments existante.

## DIGITAL CHANNELS

Parameter	Beschreibung
[Liste der verfügbaren Kanäle]	Auswahl der zu visualisierenden Elemente  Mehrfachauswahl mit Klick auf den gewünschten Eintrag in der Liste. Ausgewählte Elemente werden farbig hinterlegt dargestellt. Ein nochmaliger

Parameter	Beschreibung
	Mausklick hebt die Auswahl des Eintrags auf.
<b>Select All</b>	Sélectionne tous les éléments dans la liste.
<b>Deselect All</b>	Désactive la sélection d'éléments existante.

## AUSWAHL ANZEIGEN

Um Ihre Auswahl in der Graphen Ansicht anzuseigen, klicken Sie auf den Button **Apply**.

**Hinweis:** Ein Klick auf die seitlich vertikal angeordnete Umschaltung **COMTRADE Viewer Configuration** wechselt nur die Ansicht. Eine geänderte Auswahl der Kanäle wird dabei nicht berücksichtigt.

## 6.5.4.7 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.

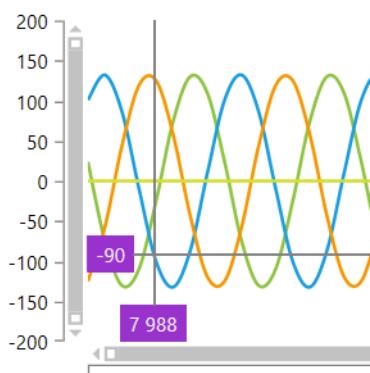
- ▶ The labeling of the digital channels corresponds to the channel description as defined in the COMTRADE file (\*.cfg).
- ▶ The colors for each channel are assigned automatically with the configured color palette.
- ▶ The time is displayed in a footer under the graph. The start time is displayed as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

## 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.

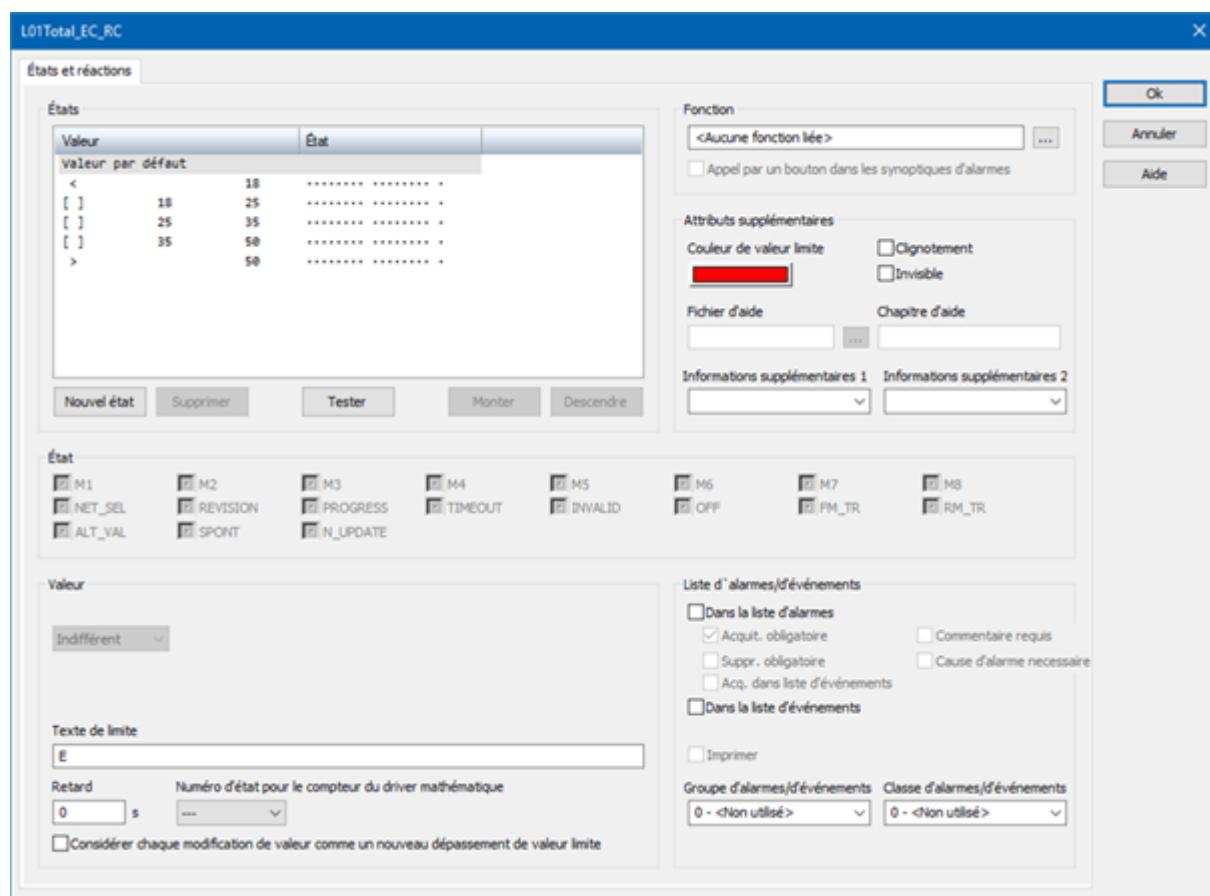
## 6.5.5 Energieklassen-Diagramm

Das Energieklassen-Diagramm, WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.



Zur Modellierung eines Energieklassen-Diagramms muss eine Reaktionsmatrix verwendet werden. Diese Reaktionsmatrix muss mit jener Variable verknüpft werden, deren Wert zur Anzeige und Einteilung in Energieklassen vorgesehen ist. Der Name der Variable muss dem Property "zVariableName" übergeben werden.

### REAKTIONS MATRIX FÜR ENERGIEKLASSEN-DIAGRAMM



The screenshot shows the configuration dialog for the 'L01Total\_EC\_RC' element. The 'Etats et réactions' tab is selected. The 'Etats' section contains a table of default values:

Valeur	Etat
<	18
[ ]	25
[ ]	35
[ ]	50
>	50

Buttons at the bottom include 'Nouvel état', 'Supprimer', 'Tester', 'Monter', and 'Descendre'. The 'Fonction' section is empty. The 'Attributs supplémentaires' section includes a color swatch for the limit value (red) and checkboxes for 'Cognottement' and 'Invisible'. The 'Informations supplémentaires' section has two dropdowns. The 'État' section lists various states: M1, NET\_SEL, ALT\_VAL, M2, REVISION, SPONT, M3, PROGRESS, N\_UPDATE, M4, TIMEOUT, M5, INVALID, M6, OFF, M7, FM\_TR, M8, RM\_TR. The 'Valeur' section shows 'Indifferent' in a dropdown. The 'Texte de limite' field contains 'E'. The 'Retard' field shows '0 s'. The 'Liste d' alarmes/d'événements' section has several checkboxes for alarm and event settings. At the bottom, there are dropdowns for 'Groupe d'alarmes/d'événements' and 'Classe d'alarmes/d'événements', both currently set to '0 - <Non utilisé>'.

Die verknüpfte Reaktionsmatrix muss folgendem Schema entsprechen:

- ▶ Der erste Zustand muss ein Bereich, oder eine „kleiner als“- Definition sein

- ▶ Anschließend können beliebig viele Bereiche definiert werden.
- ▶ Der letzte Zustand muss ein Bereich, oder eine „größer als“- Definition sein.

Für die Projektierung gilt:

1. Sollten der erste Zustand ein Bereich sein und der Wert der Variable fällt unter diesen Bereich, so wird trotzdem der erste Zustand in dem Diagramm angezeigt. Das gleiche gilt für den letzten Zustand in umgekehrter Weise.
2. Die Farben, die das WPF-Diagramm für die Klassen verwendet, sind die Grenzwertfarben, die in der Reaktionsmatrix definiert wurden.
3. Die Buchstaben für die Klassen werden mit „A“ beginnend in alphabetischer Reihenfolge gesetzt.

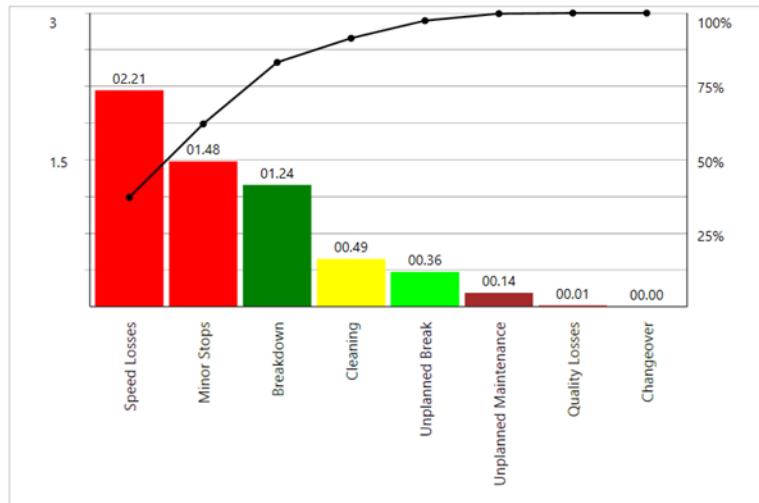
Property	Beschreibung	Wert
<b>zenonFontID</b>	ID für eine Schrift aus der ersten Schriftliste (Schriftgröße wird nicht berücksichtigt)	Integer
<b>zenonNumberOfDecimalPlaces</b>	Anzahl der angezeigten Nachkommastellen	Integer
<b>zenonVariableName</b>	Name der anzuzeigenden Variable	String

**Hinweis:** Für die Darstellung des Energieklassen-Diagramms im zenon Web Client sind zusätzliche VSTA-Programmierungen notwendig. Details dazu erhalten Sie im Kapitel Anzeige von WPF-Elementen im zenon Web Client (à la page 112).

## 6.5.6 Pareto-Diagramm

Das Pareto-Diagramm, WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.

Im Folgenden wird ein Beispiel eines Pareto-Diagrammes in der Runtime dargestellt:



Folgende Einstellungen können im WPF-Konfigurationsfenster unter **COPADATA-ELEMENT** gemacht werden:

Property	Funktion	Wert
<b>zenonBarColor1</b>	Farbe des 1. Balken	COLOR (STRING)
<b>zenonBarColor2</b>	Farbe des 2. Balken	COLOR (STRING)
<b>zenonBarColor3</b>	Farbe des 3. Balken	COLOR (STRING)
<b>zenonBarColor4</b>	Farbe des 4. Balken	COLOR (STRING)
<b>zenonBarColor5</b>	Farbe des 5. Balken	COLOR (STRING)
<b>zenonBarColor6</b>	Farbe des 6. Balken	COLOR (STRING)
<b>zenonBarColor7</b>	Farbe des 7. Balken	COLOR (STRING)
<b>zenonBarColor8</b>	Farbe des 8. Balken	COLOR (STRING)
<b>zenonBarColor9</b>	Farbe des 9. Balken	COLOR (STRING)
<b>zenonBarColor10</b>	Farbe des 10. Balken	COLOR (STRING)
<b>zenonColorPercentageLine</b>	Farbe der Prozentlinie (relative Summenhäufigkeit).	COLOR (STRING)
<b>zenonInitialisationTime</b>	Gibt an, wie lange mit dem Initialisieren des Controls gewartet wird.	INTEGER Par défaut : 150 ms
<b>zenonLineVisibility</b>	Sichtbarkeit der Prozentlinie (relative	BOOLEAN

Property	Funktion	Wert
	Summenhäufigkeit).	
<b>zenonVariable1_Label</b>	Beschriftung für den 1. Balken	<i>STRING</i>
<b>zenonVariable1_Value</b>	Wert des 1. Balken	<i>DOUBLE</i>
<b>zenonVariable2_Label</b>	Beschriftung für den 2. Balken	<i>STRING</i>
<b>zenonVariable2_Value</b>	Wert des 2. Balken	<i>DOUBLE</i>
<b>zenonVariable3_Label</b>	Beschriftung für den 3. Balken	<i>STRING</i>
<b>zenonVariable3_Value</b>	Wert des 3. Balken	<i>DOUBLE</i>
<b>zenonVariable4_Label</b>	Beschriftung für den 4. Balken	<i>STRING</i>
<b>zenonVariable4_Value</b>	Wert des 4. Balken	<i>DOUBLE</i>
<b>zenonVariable5_Label</b>	Beschriftung für den 5. Balken	<i>STRING</i>
<b>zenonVariable5_Value</b>	Wert des 5. Balken	<i>DOUBLE</i>
<b>zenonVariable6_Label</b>	Beschriftung für den 6. Balken	<i>STRING</i>
<b>zenonVariable6_Value</b>	Wert des 6. Balken	<i>DOUBLE</i>
<b>zenonVariable7_Label</b>	Beschriftung für den 7. Balken	<i>STRING</i>
<b>zenonVariable7_Value</b>	Wert des 7. Balken	<i>DOUBLE</i>
<b>zenonVariable8_Label</b>	Beschriftung für den 8. Balken	<i>STRING</i>
<b>zenonVariable8_Value</b>	Wert des 8. Balken	<i>DOUBLE</i>
<b>zenonVariable9_Label</b>	Beschriftung für den 9. Balken	<i>STRING</i>
<b>zenonVariable9_Value</b>	Wert des 9. Balken	<i>DOUBLE</i>
<b>zenonVariable10_Label</b>	Beschriftung für den 10. Balken	<i>STRING</i>
<b>zenonVariable10_Value</b>	Wert des 10. Balken	<i>DOUBLE</i>

Folgende Events können verwendet und mit zenon Funktionen verknüpft werden:

Event	Funktion	Wert
<b>zenonBar1Click</b>	Funktion, die beim Klick auf den 1. Balken ausgeführt wird.	Funktion
<b>zenonBar2Click</b>	Funktion, die beim Klick auf den 2. Balken ausgeführt wird.	Funktion

Event	Funktion	Wert
<b>zenonBar3Click</b>	Funktion, die beim Klick auf den 3. Balken ausgeführt wird.	Funktion
<b>zenonBar4Click</b>	Funktion, die beim Klick auf den 4. Balken ausgeführt wird.	Funktion
<b>zenonBar5Click</b>	Funktion, die beim Klick auf den 5. Balken ausgeführt wird.	Funktion
<b>zenonBar6Click</b>	Funktion, die beim Klick auf den 6. Balken ausgeführt wird.	Funktion
<b>zenonBar7Click</b>	Funktion, die beim Klick auf den 7. Balken ausgeführt wird.	Funktion
<b>zenonBar8Click</b>	Funktion, die beim Klick auf den 8. Balken ausgeführt wird.	Funktion
<b>zenonBar9Click</b>	Funktion, die beim Klick auf den 9. Balken ausgeführt wird.	Funktion
<b>zenonBar10Click</b>	Funktion, die beim Klick auf den 10. Balken ausgeführt wird.	Funktion

### 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> ▶ <i>None</i> ▶ <i>Automatic</i>

Property	Function	Value
		<p><i>c</i></p> <ul style="list-style-type: none"> <li>▶ Surround In</li> <li>▶ Surround Out</li> </ul>
<b>ScaleSweepAngle</b>	Angel 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> <li>▶ Rectangle</li> <li>▶ TriangleCap</li> <li>▶ Pentagon</li> <li>▶ Triangle</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>▶ Rectangle</li> <li>▶ Trapezoid</li> <li>▶ Triangle</li> </ul>
<b>MinorTicksShape</b>	Shape/type of sub ticks on the scale.	<i>Enum:</i>

Property	Function	Value
		<ul style="list-style-type: none"> <li>▶ <i>Rectangl e</i></li> <li>▶ <i>Trapezoi d</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> <li>▶ <i>Screw</i></li> </ul>
<b>PointerCapBorderBr ush</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>GaugeBackgroundBr ush</b>	Color of element background.	<i>Brush</i>
<b>PointerCapColorBrus h</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>	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: <ul style="list-style-type: none"> <li>▶ <b>Range1EndValue:</b> End value of the 1st area</li> </ul>	<i>Double</i>

Property	Function	Value
	<p>and start value of the 2nd range.</p> <ul style="list-style-type: none"> <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>	
<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> <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 72).</p>	<i>Brush</i>
<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> </ul>

Property	Function	Value
		<ul style="list-style-type: none"> <li>▶ <i>me</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 Sankey diagram

The Sankey diagram, WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.

The Sankey wizard must be used to model a Sankey diagram. The wizard creates an XML file that is then evaluated by the WPF element. To do this, the **zSankeyName** property must be given the name of the XML file. The XML file must be in the *Other* folder of a project. This is saved there by the wizard.

An example of a Sankey diagram in Runtime is shown below:

The following settings can be made in the WPF configuration window under **COPADATA-ELEMENT**:

Property	Function	Value
<b>FontSize</b>	Font size of the texts.	<i>Integer</i>
<b>zBackgroundColor</b>	Background color of the diagram.	<i>Color (String)</i>
<b>zFontColor</b>	Color of the texts.	<i>Color (String)</i>

Property	Function	Value
<b>zFontFamily</b>	Font of all texts.	<i>Font (String)</i>
<b>zLossDetectionActive</b>	Automatic loss detection activated/deactivated. If <i>true</i> , then losses are automatically shown at a node points as flows.	<i>Bool</i>
<b>zNoDataText</b>	Text that is displayed if there are no values to display and <b>zPreviewActive</b> is <i>false</i> .	<i>String</i>
<b>zNoValidXMLText</b>	Text that is displayed if no valid XML file with entered name has been found and <b>zPreviewActive</b> is false.	<i>String</i>
<b>zNumberOfDecimalPlaces</b>	Denotes how many decimal places are to be displayed.	<i>Integer</i>
<b>zPreviewActive</b>	Display of a preview activated/deactivated.  The preview can be displayed if  There is no data present (the modeled diagram is filled with default values) or  the XML file was not found or  this does not contain a valid definition (an example Sankey diagram is displayed).	<i>Bool</i>
<b>zRefreshRate</b>	Rate at which the diagram is refreshed in ms.	<i>Integer</i>
<b>zSankeyName</b>	Name of the XML file with the modeling of the diagram.	<i>String</i>
<b>zShowRelativeValues</b>	Display of the values in absolute <i>false</i> or relative values <i>true</i> .	<i>Bool</i>

**Note:** Additional VSTA programming is necessary for the display of the Sankey diagrams 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 112).

### 6.5.9 Temperaturanzeige - TemperatureIndicatorControl

Property	Funktion	Wert
<b>CurrentValue</b>	Aktueller Wert, der angezeigt werden soll.	<i>Double</i>
<b>MinValue</b>	Minimalwert der Skala.	<i>Double</i>

Property	Funktion	Wert
<b>.MaxValue</b>	Maximalwert der Skala.	<i>Double</i>
<b>MajorTicksCount</b>	Anzahl der Hauptticks der Skala.	<i>Integer</i>
<b>MinorTicksCount</b>	Anzahl der Nebenticks der Skala.	<i>Integer</i>
<b>TickNegativColor</b>	Farbe des negativen Hauptticks (verlaufend zu TickPositivColor).	<i>Color</i>
<b>TickPositivColor</b>	Farbe des positiven Hauptticks (verlaufend zu TickNegativColor).	<i>Color</i>
<b>MinorTickCount</b>	Farbe der Nebenticks.	<i>Color</i>
<b>ElementBorderBrush</b>	Farbe des Elementrahmens.	<i>Brush</i>
<b>ElementBackgroundBrush</b>	Farbe des Elementhintergrunds.	<i>Brush</i>
<b>ElementGlasReflection</b>	Aktivierung des Glaseffektes auf dem Element.	<i>Visibility</i>
<b>ElementFontFamily</b>	Elementschriftart.	<i>Font</i>
<b>IndicatorColor</b>	Farbe der Indikatorfüllfarbe.	<i>Color</i>
<b>IndicatorBorderColor</b>	Farbe des Indikatorrahmens.	<i>Color</i>
<b>MajorTickSize</b>	Größe der Hauptticks der Skala.	<i>Size</i>
<b>MinorTickSize</b>	Größe der Nebenticks der Skala.	<i>Size</i>
<b>ScaleLetteringDistance</b>	Abstand der Skalenbeschriftung (vertikal), jeder x-Haupttick soll beschriftet werden.	<i>Integer</i>
<b>IndicatorScaleDistance</b>	Abstand zwischen Indikator und Skala (horizontal).	<i>Double</i>
<b>ScaleFontSize</b>	Schriftgröße der Skala.	<i>Double</i>
<b>ScaleFontColor</b>	Schriftfarbe der Skala.	<i>Color</i>
<b>Unit</b>	Einheit.	<i>String</i>
<b>ElementStyle</b>	Form/Art des Elements.	<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> <ul style="list-style-type: none"> <li>▶ <i>None</i></li> <li>▶ <i>Automatic</i></li> <li>▶ <i>SurroundIn</i></li> <li>▶ <i>SurroundOut</i></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>ElementStyle</b>	Display type of the element	<i>Enum:</i> <ul style="list-style-type: none"> <li>▶ <i>Knob</i></li> <li>▶ <i>Plate</i></li> <li>▶ <i>None</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>

Property	Function	Value
<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>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>

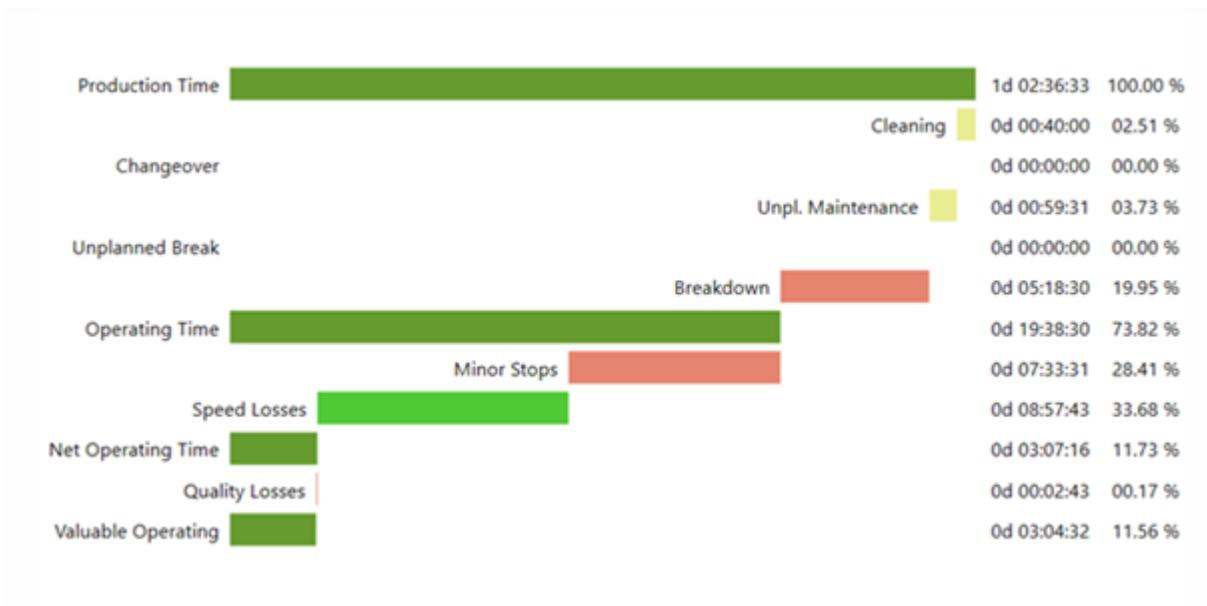
Property	Function	Value
<b>ValueVisibility</b>	Activating the value display.	<i>Boolean</i>

## 6.5.11 Diagramme en cascade

Le **diagramme cascade** WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.

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 :



**Remarque :** Cette capture d'écran est uniquement disponible en anglais.

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>zenonWaterfallIdentifie</b>	Nom du diagramme en	<i>String</i>

Propriété	Fonction	Valeur
r	cascade.	
<b>zenonZSystemModel</b>	Groupe d'équipements des variables utilisées.	<i>String</i>

**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 112).

## 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 un affichage dans le Runtime, les barres, l'intitulé et l'affichage des valeurs pour une exécution d'un clic de la souris 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. Nommez la fonction dans la propriété **Nom**.

**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*), est utilisé.

## 6.5.12 BACnet WPF Control

The **BACnet Schedule Control** WPF-Element steht Partnern von COPA-DATA zur Verfügung und ist für diese über die COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>) erhältlich.

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

Conditions requises :

- ▶ Vous devez être un partenaire COPA-DATA enregistré.
- ▶ Vous devez d'ores et déjà avoir accès au site Web COPA-DATA Partner Community.

## TÉLÉCHARGEZ UN ÉLÉMENT WPF À PARTIR DU PORTAIL PARTENAIRES COPA-DATA

Exécutez les étapes suivantes pour télécharger l'élément WPF :

1. Ouvrez COPA-DATA Partner Community (<https://www.copadata.com/en-us/partner-community/>).
2. Connectez-vous dans la zone appelée **MY AREA**.
3. Entrez « Partenaire » en tant que terme de recherche à côté de la loupe. **COPA-DATA Partner Community** apparaît avec un lien dans les résultats.
4. Cliquez sur le lien.
5. Dans **Partner Login Area**, cliquez le bouton **MORE**.  
L'onglet **Overview** est ouvert.
6. Basculez vers l'onglet **Documents & Downloads**.
7. Sous **SELECT CATEGORY**, activez la case **Élément WPF**.

8. Cliquez sur **Search**.

L'intitulé est masqué.

Un cadre orange entoure le champ de recherche.

9. Cliquez dans le cadre et entrez **Bacnet**.

10. Le **BACNetWPFScheduler** est affiché.

11. Sous **Actions**, cliquez sur le symbole orange pour télécharger le **BACnet Schedule Control**.

12. Enregistrez le dossier localement.

13. Décompressez le fichier.

14. Ouvrez le dossier contenant les fichiers *BacnetSchedulerControl.xaml* et *BacnetSchedulerWPF.dll*.

## IMPORTEZ BACNET SCHEDULE CONTROL DANS L'EDITOR ZENON

Pour créer un élément WPF

1. Ouvrez zenon Editor

2. Accédez au nœud **Fichiers** dans le gestionnaire de projets.

3. Étendez la vue de ce nœud en cliquant sur le bouton [+].  
fenêtre avec l'arborescence est ouverte et les sous-nœuds sont affichés.

4. Allez au sous-nœud **Éléments graphiques**.

- a) Sélectionnez l'entrée **Importer le fichier** dans la barre d'outils de la vue détaillée.  
La boîte de dialogue de sélection de fichiers s'ouvre.

- b) Sélectionnez le fichier intitulé *BacnetSchedulerControl.xaml*.

- c) Confirmez la sélection en cliquant sur le bouton **Ouvrir**.

5. Allez au sous-nœud **Divers**.

- a) Sélectionnez l'entrée **Importer le fichier** dans la barre d'outils de la vue détaillée.  
La boîte de dialogue de sélection de fichiers s'ouvre.

- b) Sélectionnez le fichier intitulé *BacnetSchedulerWPF.dll*.

- c) Confirmez la sélection en cliquant sur le bouton **Ouvrir**.

### 6.5.12.2 Configuration dans zenon Editor

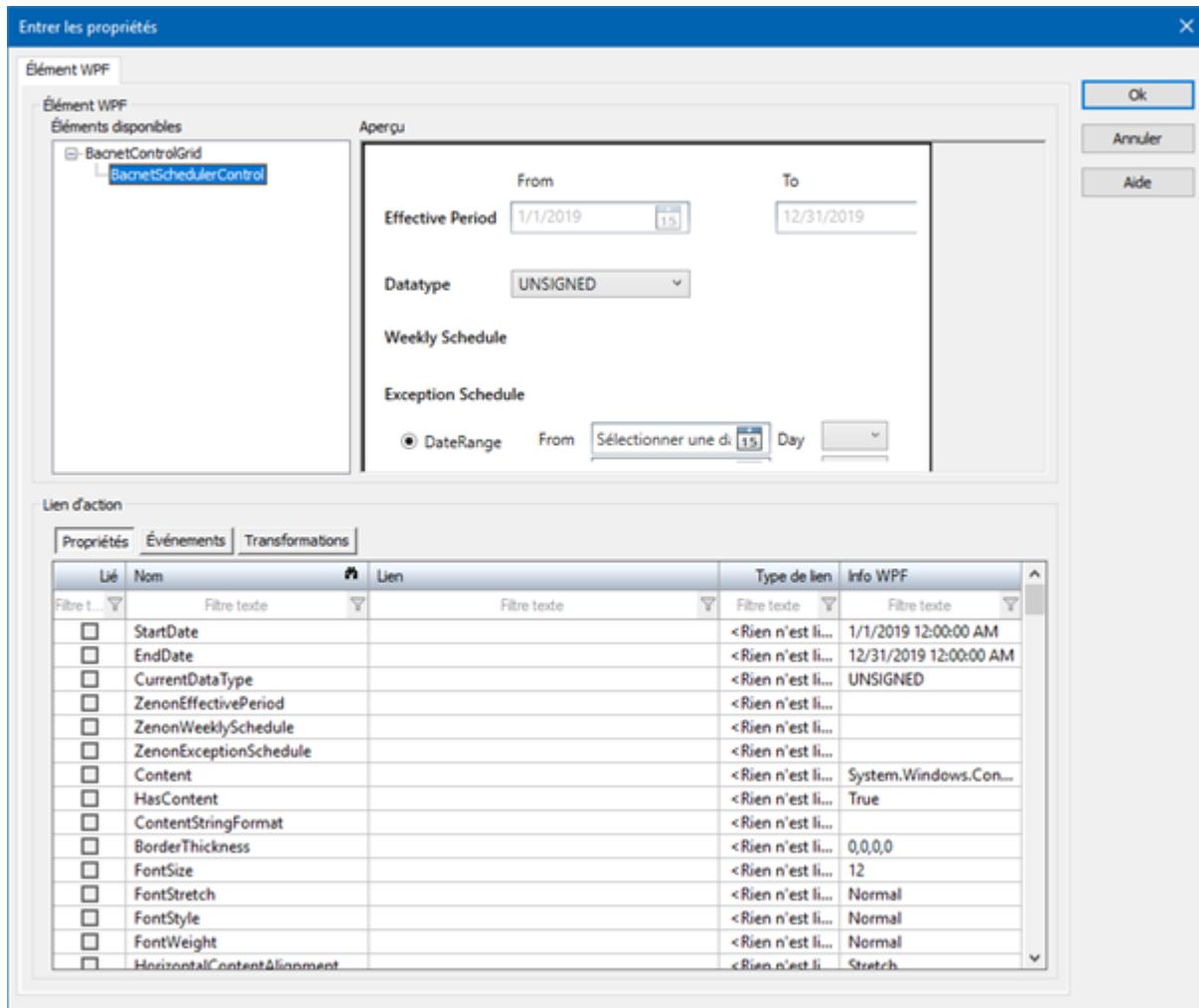
Exécutez les étapes suivantes pour utiliser le **BacnetScheduler** dans zenon :

1. Créez un nouveau synoptique.

Pour ce faire, sélectionnez la commande **Nouveau** dans la barre d'outils ou dans le menu contextuel du nœud des **Synoptiques**.

2. Modifiez les propriétés du synoptique :
  - a) Nommez le synoptique dans la propriété **Nom**.
  - b) Sélectionnez le type de synoptique de votre choix dans la propriété **Type de synoptique** (type de synoptique recommandé : *Standard*).
  - c) Sélectionnez le cadre souhaité dans la propriété **Gabarit**.
3. Configurez le contenu du synoptique :
  - a) Ouvrez le synoptique dans lequel l'élément WPF doit être inséré.
  - b) Dans la barre d'outils **Éléments**, sélectionnez élément WPF (**WPF**) .
  - c) Faites glisser l'élément WPF dans le synoptique à l'aide du bouton gauche de la souris.
  - d) La fenêtre **Sélection de fichier** s'ouvre.
  - e) Sélectionnez le fichier **BacnetSchedulercontrol.xaml** et confirmez l'entrée en cliquant sur **OK**.
  - f) L'élément WPF est ajouté.
4. Ajustez la taille de l'élément WPF à votre synoptique zenon.
  - a) Sélectionnez l'élément WPF de votre choix.
  - b) Accédez au groupe de propriétés **Position**.
  - c) Définissez les paramètres pour la largeur de l'élément à l'aide de la propriété **Largeur [pixels]** et la hauteur avec la propriété **Hauteur [pixels]**.  
Taille minimale recommandée : 920 x 920 pixels

### 6.5.12.3 Konfigurierbare Control-Eigenschaften



Führen Sie folgende Schritte aus, um Ihr BACnet WPF Control zu parametrieren:

1. Markieren Sie das WPF-Element.
2. Wechseln Sie in die Eigenschaftengruppe **Liens WPF**.
3. Klicken Sie bei der Eigenschaft **Configuration** die Schaltfläche ...
4. Der Dialog Element-Eingabe wird geöffnet.
5. Erweitern Sie den Knoten **BacnetControlGrid** im Bereich **Vorhandene Elemente**. Im Knoten wird der Eintrag **BacnetSchedulerControl** sichtbar.
6. Klicken Sie den Eintrag **BacnetSchedulerControl**. Die Ansicht im Bereich **Aktionsverknüpfungen** wird aktualisiert.
7. Parametrieren Sie die WPF-Inhalte:
  - a) Klicken Sie einmalig in der jeweiligen Zeile in der Spalte **Verknüpfungsart**. Eine Dropdownliste wird aufgeschaltet.

- b) Wählen Sie in der Dropdownliste die entsprechende Verknüpfungsart.
  - c) Durch Klick in der Spalte **Verknüpfung** wird der Dialog **Element-Eingabe** aufgeschaltet. Dieser ist an die projektierte Verknüpfungsart angepasst (z.B. Auswahl einer Variable, Eingabe eines Wertes, ...)
8. Bestätigen Sie Ihre Parametrierung mit Klick auf **OK**.

## MÖGLICHE ARTEN DER VERKNÜPFUNG

Es können eine oder mehrere Parameter für dieses WPF verknüpft werden:

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

### Informations

Die Parametrierung des WPF Elements wird im zenon in der Eigenschaftengruppe **Liens WPF** des WPF-Elements visualisiert. Dabei wird jede WPF-Verknüpfung mit einem eigenen Unterknoten oder Eigenschaftenüberschrift angezeigt.

## 6.5.12.4 Runtime view

**Effective Period**

From:

To:

Use Timespan

**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**

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
Effective Period	<input style="width: 100px; height: 25px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px;" type="text" value="1/1/2018"/> 	<input style="width: 100px; height: 25px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px;" type="text" value="12/31/2018"/> 
		<input style="width: 100px; height: 30px; background-color: #5577AA; color: white; border: none; border-radius: 5px; font-weight: bold; font-size: 10pt;" type="button" value="Save"/>

In this area, the value of the variable that has been linked in configuration of the control (à la page 99) 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>: The Effective Period is not used.</li> <li>▶ <i>Active</i>: 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> <p>Par défaut : 31. 12. 2018</p> <p><b>Note:</b> Only active if the <b>Use Timespan</b> option is active.</p>
<b>Save</b>	<p>Writes the current configuration for the <b>Effective Period</b> to the PLC.</p>

## 6.5.12.4.2 Runtime view

### Weekly Schedule

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

In this area, the value of the variable that has been linked in configuration of the control (à la page 99) in the Editor for the **ZenonWeeklySchedule** option is displayed.

Parameters	Description
[List of the content of the loaded Weekly Schedules]	<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> Selection of the day from a drop-down list.</li> <li>▶ <i>Time</i> Entry of the time Input format: <i>hh:mm:ss:ms</i></li> <li>▶ <i>Value</i> Value of the respective schedule. The value represents that of the data type:  <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> </ul> </li> </ul>

Parameters	Description
	<p>[8] = BitString  [9] = Enumerated  [10] = Date  [11] = Time  [12] = BACnetObjectIdentifier</p> <p>Default: [0]</p> <p><b>Note:</b> The data type is required in the <i>Values</i> parameter when configuring. The data type is given as a number according to the top list. Note that this must be given in square brackets []. Example: [1] for BOOLEAN.</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:  [1] TRUE  For Boolean with value TRUE</p> <p>[10] 2/3/2021  For data type date with value February 3, 2021  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 without a request for confirmation.</p>
<b>New</b>	<p>Creates a new <i>Weekly Schedule</i>.</p>
<b>Delete</b>	<p>Deletes the selected entry from the <b>Weekly Schedule</b> list. 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>
<b>Save</b>	<p>Writes the current configuration for the <b>Weekly Schedules</b> to the PLC.</p>

## 6.5.12.4.3 Runtime Ansicht

**Exception Schedule**

DateRange      From   Day    
 SingleDate      To   Day  Priority 
  
 WeekNDay   
  
 CalenderRef

Event Type	Period	Priority

**Time values**

Time	Value

In diesem Bereich wird der Wert jener Variable angezeigt, die bei der Konfiguration des Controls (à la page 99) im Editor bei der Option **ZenonExceptionSchedule** verknüpft wurde.

Parameter	Beschreibung
Typ des Exception Schedules	<p>Typ des Zeitplans für Ausnahmen (Exception Schedule).</p> <p>Auswahl aus Optionenbox:</p> <ul style="list-style-type: none"> <li>▶ <i>DateRange</i> Der Exception Schedule ist mit einem Zeitbereich definiert. Der Zeitbereich wird mit den Optionen <b>From</b>, <b>Day</b>, <b>To</b> und <b>Day</b> zu projektiert.</li> <li>▶ <i>SingleDate</i> Der Exception Schedule ist mit einem exakten Tag definiert. Der Tag ist mit den Optionen <b>Single</b>, und <b>Day</b> zu projektieren.</li> <li>▶ <i>WeekNDay</i> Der Exception Schedule ist mit mehreren</li> </ul>

Parameter	Beschreibung
	<p>Tag definiert. Diese Tage werden in den Dropdownlisten für <b>WeekNDay</b> projektiert.</p> <ul style="list-style-type: none"> <li>▶ <i>CalenderRef</i> Der Exception Schedule ist mit einer Referenz auf ein bestehendes BACNet Kalenderobjekt definiert.</li> </ul>
<b>From</b> (bei Eventtype DateRange)  <b>Single</b> (bei Eventtype SingleDate)	<p>Beginn des Eventtype <i>Exception Schedules</i>. Eingabe des Datums. Eingabe via manueller Eingabe oder Auswahl aus Kalender. Der Kalender wird durch Klick auf das Kalendersymbol als Dropdowndialog angezeigt.</p> <p>Par défaut : <i>vide</i> (<i>Select a date</i>)</p> <p><b>Hinweis:</b> Nur aktiv, wenn die Optionen <b>DateRange</b> oder <b>SingleDate</b> aktiv sind.</p>
<b>Day</b> (in Zeile <b>From</b> )	<p>Wochentag des Beginns des Eventtype <i>Exception Schedules</i>.</p> <p>Wird ein Datum in der Option <b>From</b> projektiert wird der entsprechende Wochentag automatisch für diese Option gesetzt.</p> <p>Sélection du jour de la semaine à partir d'une liste déroulante :</p> <ul style="list-style-type: none"> <li>▶ <i>Mon</i> Lundi</li> <li>▶ <i>Tue</i> Mardi</li> <li>▶ <i>Wed</i> Mercredi</li> <li>▶ <i>Thu</i> Jeudi</li> <li>▶ <i>Fri</i> Vendredi</li> <li>▶ <i>Sat</i> Samedi</li> <li>▶ <i>Sun</i></li> </ul>

Parameter	Beschreibung
	<p>Dimanche</p> <p>Par défaut : <i>vide</i></p> <p><b>Hinweis:</b> Nur aktiv, wenn die Optionen <b>DateRange</b> oder <b>SingleDate</b> aktiv sind.</p>
<b>To</b>	<p>Ende des Eventtype <i>Exception Schedules</i>. Eingabe des Datums.</p> <p>Eingabe via manueller Eingabe oder Auswahl aus Kalender. Der Kalender wird durch Klick auf das Kalendersymbol als Dropdowndialog angezeigt.</p> <p>Par défaut : <i>vide</i> (<i>Select a date</i>)</p> <p><b>Hinweis:</b> Nur aktiv, wenn die Option <b>DataRange</b> aktiv ist.</p>
<b>Day</b> (in Zeile <b>To</b> )	<p>Wochentag des Endes des Eventtype <i>Exception Schedules</i>.</p> <p>Wird ein Datum in der Option <b>To</b> projektiert wird der entsprechende Wochentag automatisch für diese Option gesetzt.</p> <p>Sélection du jour de la semaine à partir d'une liste déroulante :</p> <ul style="list-style-type: none"> <li>▶ <i>Mon</i> Lundi</li> <li>▶ <i>Tue</i> Mardi</li> <li>▶ <i>Wed</i> Mercredi</li> <li>▶ <i>Thu</i> Jeudi</li> <li>▶ <i>Fri</i> Vendredi</li> <li>▶ <i>Sat</i> Samedi</li> <li>▶ <i>Sun</i> Dimanche</li> </ul>

Parameter	Beschreibung
	<p><b>Hinweis:</b> Nur aktiv, wenn die Option <b>DataRange</b> aktiv ist.</p>
<b>Priority</b>	<p>Priorität für den Eventtype laut BACnet Standard. Auswahl aus Dropdownliste: Werte 1 - 16</p>
<b>WeekNDay</b>	<p>Auswahl des Datums für den Eventtype <i>Exception Schedule</i>. aus Dropdownlisten.</p> <ul style="list-style-type: none"> <li>▶ Dropdownliste 1: Auswahl des Monats <b>Hinweis:</b> Der Eintrag <i>any month of year</i> entspricht jedem Monat.</li> <li>▶ Dropdownliste 2: Auswahl des Tages nach vorgegebenen Kriterien. Dropdownliste 3: Auswahl des Wochentages <b>Hinweis:</b> Der Eintrag <i>any day of week</i> entspricht jedem Wochentag einer Woche.</li> </ul> <p><b>Hinweis:</b> Nur aktiv, wenn die Option <b>WeekNDay</b> aktiv ist.</p>
<b>CalenderRef</b>	<p>Referenz auf ein bestehendes BACnet Objekt vom Typ <b>CalendarObject</b>.</p> <p>Eingabe der numerischen <i>Object Instance</i> des BACnet Objekts. Laut BACnet Standard ist dies immer eine siebenstellige Bitnummer.</p> <p>Gültiger Eingabebereich (laut BACnet Standard): 0 - 4194302</p> <p><b>Hinweis:</b> Um dies zu erreichen, muss die hier im Control eingegebenen <i>Object Instance</i>-Nummer mit vorangestellten 0 aufgefüllt werden.</p> <p><b>Beispiel:</b> <b>CalendarObject</b> <i>Instance Nummer = 1</i> Eingabe in Control = 0000001</p>
[Liste der Inhalte des geladenen	<ul style="list-style-type: none"> <li>▶ Der Zeitplan wird in der Liste <b>Exception</b></li> </ul>

Parameter	Beschreibung
<b>Exception Schedules]</b>	<p><b>Schedule</b> als Listeneintrag angezeigt. Die Parametrierung erfolgt mit den oben aufgelisteten Optionen:</p> <ul style="list-style-type: none"> <li>▶ <i>Event Type</i>: Entspricht der Auswahl des Event Types aus der Optionenbox.</li> <li>▶ <i>Period</i>: Darstellung des Datums. Je nach gewähltem Eventtyp kann die Darstellung variieren: [Date] - [Date]</li> <li>▶ <i>Priority</i>: Projektete Priorität</li> </ul>
<b>Load</b>	<p>Liest die aktuelle Konfiguration für die <b>Exception Schedules</b> auf die SPS.</p> <p>Der geladenen <i>Exception Schedule</i> wird in der Liste angezeigt.</p> <p><b>Hinweis:</b> Bei einem <b>Load</b> werden alle ungesicherten Konfigurationen in der Runtime ohne Rückfrage durch die Werte der SPS ersetzt.</p>
<b>New</b>	<p>Übernimmt die Projektierung aus dem Bereich <b>Exception Schedule</b> und legt einen neuen Eintrag in der Liste an.</p>
<b>Delete</b>	<p>Löscht den gewählten Eintrag aus der Liste <b>Exception Schedule</b>. Der Eintrag wird ohne vorherige Sicherheitsabfrage sofort gelöscht.</p> <p>Mehrfachauswahl ist möglich, wird jedoch bei Klick auf <b>Delete</b> ignoriert.</p>
<b>Save</b>	<p>Schreibt die aktuelle Konfiguration für die <b>Exception Schedules</b> auf die SPS.</p>

## TIME VALUES

Zeitwerte für einen projektierte Exception Schedule Event.

Parameter	Beschreibung
[Liste der projektieren Time values für das	<ul style="list-style-type: none"> <li>▶ <i>Time</i></li> </ul>

Parameter	Beschreibung
ausgewählte Exception Schedule Event]	<p>Eingabe der Zeit Eingabeformat: <i>hh:mm:ss:ms</i></p> <ul style="list-style-type: none"> <li>▶ <i>Value</i> Wert des jeweiligen Zeitplans. Der Wert repräsentiert den des Datenntyp:  <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>Hinweis:</b> Bei der Projektierung der Datentyp zwingend im Parameter <i>Values</i> erforderlich. Der Datentyp wird entsprechend der oberen Liste als Zahl angegeben. Beachten Sie, dass diese Angabe zwingend in eckigen Klammern [] angegeben werden muss. Beispiel: [1] für <i>BOOLEAN</i>.</p> <p>Zusätzlich zum Datentyp kann auch der entsprechende Wert des Datentyps projektiert werden. Dieser Wert wird nach dem Datentyp eingegeben.</p> <p>Beispiele:  [1] <i>TRUE</i>  für Boolean mit Wert TRUE</p> <p>[10] <i>2/3/2021</i>  für Datentyp Datum mit Wert 3. Februar 2021  Datumseingabe im BACnet Standard-Format.</p>
<b>New</b>	Erstellt für den ausgewählten Exception Schedule Event einen neuen Time value Eintrag.
<b>Delete</b>	Löscht den ausgewählten Time value Eintrag. Der

Parameter	Beschreibung
	<p>Eintrag wird ohne vorherige Sicherheitsabfrage sofort gelöscht.</p> <p>Mehrfachauswahl ist möglich, wird jedoch bei Klick auf <b>Delete</b> ignoriert.</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 Affichage d'éléments WPF dans zenon Web Client

Pour pouvoir également utiliser les éléments WPF prêts à l'emploi "**Diagramme de classe énergétique**", "**Diagramme Sankey**" et "**Diagramme en cascade**" pour l'affichage dans zenon Web Client, des modifications doivent être apportées au projet :

- ▶ Configuration dans l'Editor zenon (à la page 112)
- ▶ Adapter le code VSTA (à la page 112)

### 6.6.1 Projektierung im zenon Editor

Führen Sie im zenon Editor folgende Projektierungsschritte aus, um bestimmte WPF-Elemente auch im zenon Web Client anzeigen zu können:

#### WPF IN ZENON BILD PLATZIEREN:

- ▶ Platzieren Sie das Element WPF in einem zenon Bild.
- ▶ Vergeben Sie in der Eigenschaft **ID d'élément** einen eindeutigen Namen.  
Sie finden diese Eigenschaft in der Eigenschaftengruppe **Général**.
- Hinweis:** Wurde der Name für ein Element im Bild bereits in einem anderen Element vergeben, erscheint ein Warndialog.
- ▶ Verwenden Sie den hier vergebenen Elementnamen im VSTA-Code.

### 6.6.2 Code VSTA (complexe)

Pour ajouter le code de programmation pour l'affichage d'éléments WPF dans zenon Web Client, suivez ces instructions :

1. Dans zenon Editor, accédez au noeud **Interfaces de programmation**.
2. Sélectionnez le noeud **VSTA**, puis cliquez avec le bouton droit sur **Ouvrir l'éditeur VSTA avec le module complémentaire du projet....**
3. La boîte de dialogue de création d'un projet VSTA s'affiche à l'écran.
4. Sélectionnez l'entrée C# dans la boîte de dialogue **Créer un nouveau projet VSTA**.
5. Créez (ou copiez) le code ci-dessous.

6. Insérez le nom de l'élément WPF dans le code.

**Remarque :** Lors de l'ouverture de l'éditeur VSTA, notez si le code suivant est déjà présent dans la configuration du projet. Pour l'affichage de l'élément WPF dans Web Client, comparez le code existant et effectuez toute modification nécessaire. Veuillez consigner les commentaires à ce sujet dans le code échantillon.

## CODE VSTA

```
//En tant que membre :  
zenOn.IDynPictures zScreens = null;  
  
string[] WPFElements = {"WPF_Control", "WPFWebclient_1", "WPFWebclient_2"}; //Noms des éléments de synoptique WPF qui apparaissent dans le projet zenon et nécessitant un accès à l'API (autant que vous le souhaitez)  
  
//Ajouter les trois lignes de code suivantes dans la fonction d'archive de projet :  
void ThisProject_Active()  
{  
    zScreens = this.DynPictures();  
    zScreens.Open += new zenOn.DDynPicturesEvents_OpenEventHandler(zScreens_Open);  
    zScreens.Close += new zenOn.DDynPicturesEvents_CloseEventHandler(zScreens_Close);  
}  
//Ajouter les deux lignes de code suivantes dans la fonction inactive du projet :  
void ThisProject_Inactive()  
{  
    zScreens.Open -= new zenOn.DDynPicturesEvents_OpenEventHandler(zScreens_Open);  
    zScreens.Close -= new zenOn.DDynPicturesEvents_CloseEventHandler(zScreens_Close);  
  
    //Version finale et collecte des déchets de tout objet API.  
    FreeObjects();  
}  
  
//Ajouter deux nouveaux gestionnaires d'évènement :  
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 Code VSTA (simplifié)

Si un seul élément WPF est utilisé dans un synoptique zenon, le code simplifié suivant peut également être utilisé. Pour cela, les noms de l'élément WPF et du synoptique dans lequel il est utilisé doivent être insérés. Ce code est ensuite recommandé si, pour chaque projet, un seul des éléments WPF prêts à l'emploi est utilisé.

#### CODE VSTA

```

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

//Ajouter à la fonction active du projet :
void ThisProject_Active()
{
    zScreen = this.DynPictures().Item(wpfPicture);
    zScreen.Open += new zenOn.OpenEventHandler(zScreen_Open);
    zScreen.Close += new zenOn.CloseEventHandler(zScreen_Close);
}

//Ajouter à la fonction inactive du projet :
void ThisProject_Inactive()

```

```
{  
zScreen.Open -= new zenOn.OpenEventHandler(zScreen_Open);  
zScreen.Close -= new zenOn.CloseEventHandler(zScreen_Close);  
  
//Version finale et collecte des déchets de tout objet API.  
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 Exemples : Intégration de WPF dans zenon

Les exemples suivants décrivent la création de fichiers XAML et leur intégration sous forme d'éléments WPF dans zenon :

- ▶ Intégration d'un bouton sous forme d'élément WPF XAML dans zenon (à la page 120)
- ▶ Intégration d'un bargraphe sous forme d'élément WPF XAML dans zenon (à la page 116)
- ▶ Intégration d'un contrôle DataGrid dans zenon (à la page 127)

## 6.7.1 Intégration d'un bargraphe sous forme d'élément WPF XAML dans zenon

Exemple de structure :

- ▶ Création d'un bargraphe (à la page 16) dans Adobe Illustrator et conversion au format WPF
- ▶ Intégration dans zenon
- ▶ Liaison avec des variables
- ▶ Adaptation du bargraphe à l'élément WPF

### CRÉER LE BARGRAPHE

La première étape consiste à générer un bargraphe conformément aux indications fournies au chapitre Flux de travail avec Adobe Illustrator (à la page 16). Pour pouvoir utiliser le fichier XAML dans zenon, insérez ceci dans l'arborescence du projet dans le dossier **Fichiers/graphiques**.

### INTÉGRATION D'UN BARGRAPHE

**Remarque :** Un projet zenon doté du contenu suivant est utilisé pour la description suivante :

- ▶ Un synoptique vide en tant que synoptique de départ
- ▶ Quatre variables du driver interne pour :
  - ▶ Échelle 0
  - ▶ Échelle *centrale*
  - ▶ Échelle *élevée*
  - ▶ Valeur actuelle
- ▶ Une variable provenant du driver mathématique permettant d'afficher la valeur actuelle (255).

Pour intégrer le bargraphe :

1. Ouvrez le synoptique vide.
2. Insérez un **élément WPF** (à la page 49) dans le synoptique.
3. Sélectionnez **Fichier XAML** dans la fenêtre des propriétés.
4. Sélectionnez le fichier XAML souhaité (par exemple, **bar graph\_vertical.xaml**) et fermez la boîte de dialogue.

## AJUSTER LE BARGRAPHE

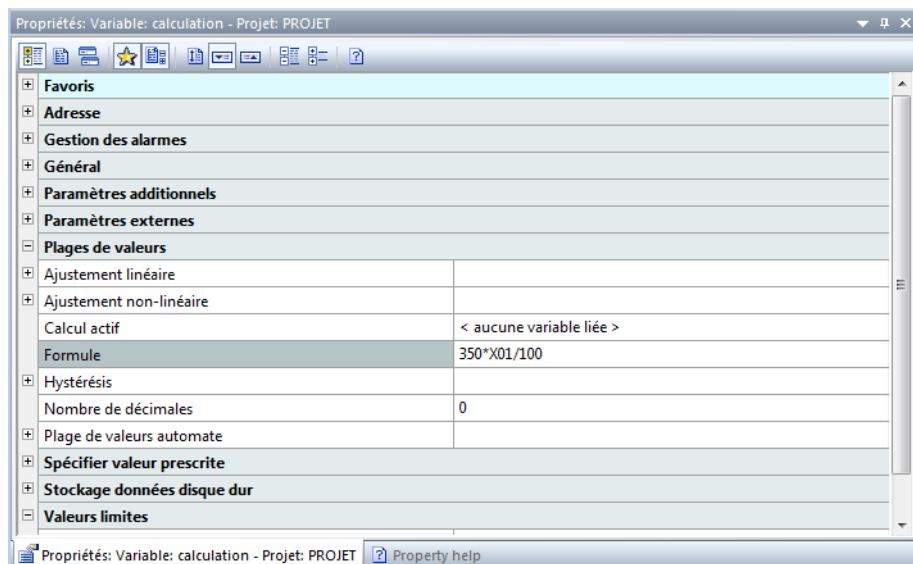
Avant la configuration, l'échelle du fichier XAML doit être adaptée, si nécessaire :



Pour cela :

- ▶ Créez une nouvelle variable mathématique qui calcule la nouvelle valeur par rapport à l'échelle, par exemple :
- ▶ Variable : 0-1000

- ▶ Variable mathématique {valeur créée dans le fichier xaml}\*Variable/1000

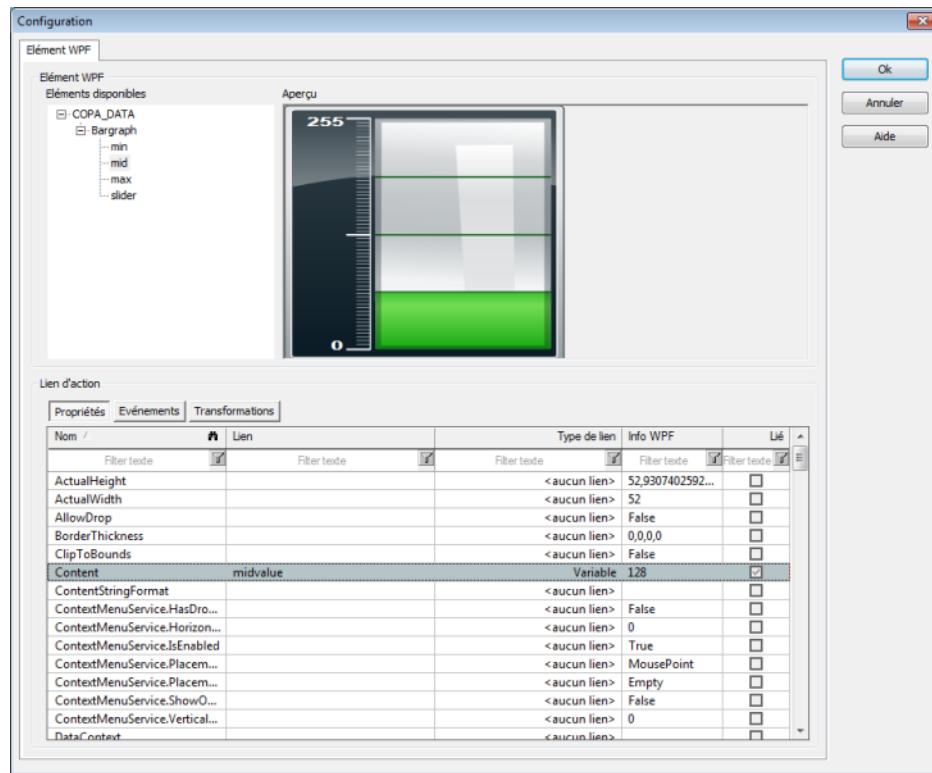


Le fichier XAML est alors configuré.

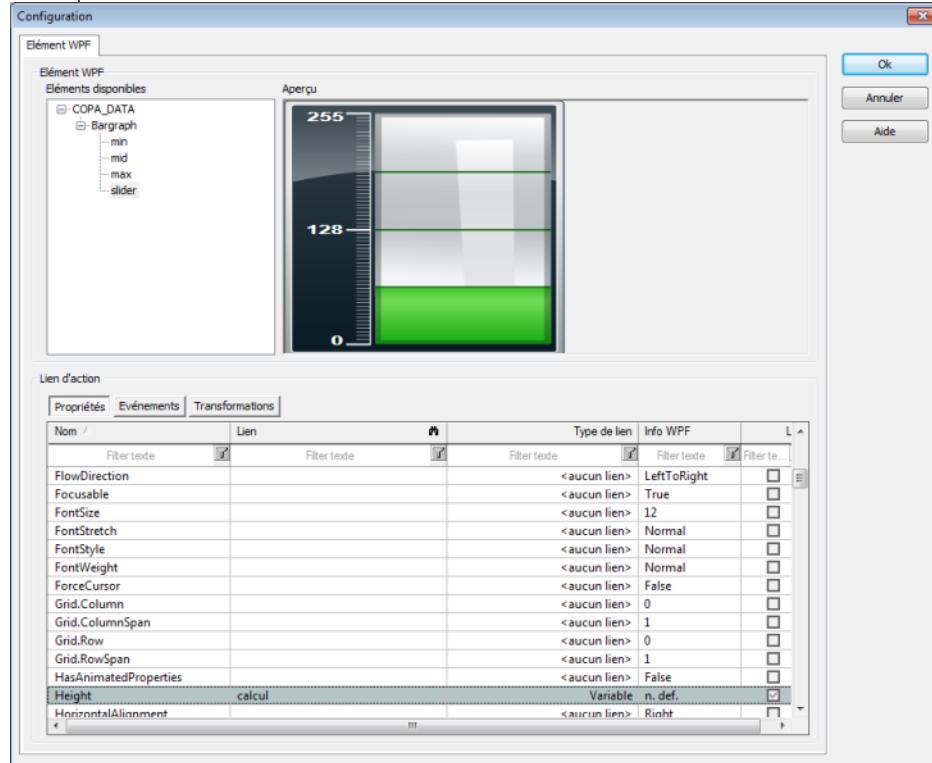
## CONFIGURER LE BARGRAPHE

1. Cliquez sur l'élément WPF et sélectionnez la propriété **Configuration**.
2. La boîte de dialogue de configuration affiche un aperçu du fichier XAML sélectionné.

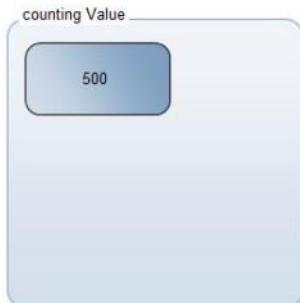
3. Sélectionnez la valeur minimum, la valeur moyenne et la valeur maximum et liez chacune de ces valeurs à la variable correspondante dans la propriété **Contenu**.



4. Sélectionnez la **réglette** et liez la propriété **Valeur** aux variables mathématiques (dans notre exemple : calcul)



5. Contrôlez le projet de planification dans le Runtime :



### 6.7.2 Intégration d'un bouton sous forme d'élément WPF XAML dans zenon

Exemple de structure :

- ▶ Création d'un bouton (à la page 12) dans Microsoft Expression Blend
- ▶ Intégration dans zenon
- ▶ Lien à une variable et une fonction
- ▶ Ajustement du bouton à la taille de l'élément
- ▶ Créez le bouton

Lors de la première étape, créez un bouton conformément aux indications fournies au chapitre Créer un bouton sous forme de fichier XAML dans Microsoft Expression Blend (à la page 12). Pour pouvoir utiliser le fichier XAML dans zenon, insérez ceci dans l'arborescence du projet dans le dossier

**Fichiers/graphiques.**

## INTÉGRER LE BOUTON

**Remarque :** Un projet zenon doté du contenu suivant est utilisé pour la description suivante :

- ▶ Un synoptique vide en tant que synoptique de départ

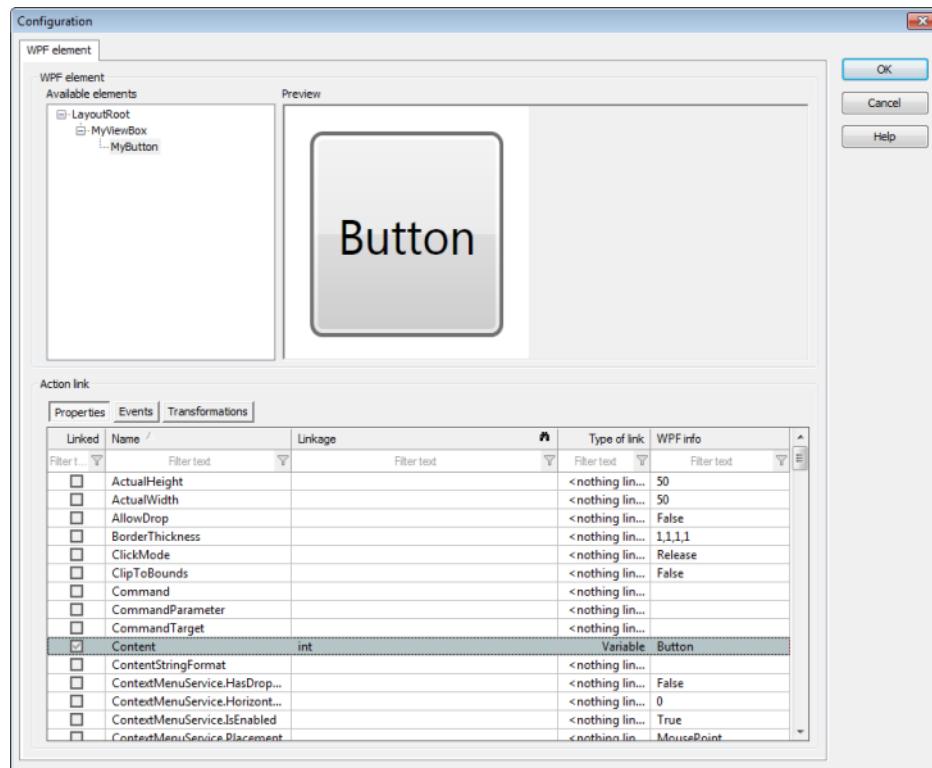
- ▶ Une variable interne **int** de type **Int**
- ▶ Une fonction **Fonction\_0** de type **Ecrire/modifier valeur prescrite** avec :
  - ▶ L'option **Directement sur matériel** activée
  - ▶ Les valeurs cible doivent être configurées avec la valeur **45**.

Pour intégrer le bouton :

1. Ouvrez le synoptique vide.
2. Insérez un **élément WPF** (à la page 49) dans le synoptique.
3. Sélectionnez **Fichier XAML** dans la fenêtre des propriétés.
4. Sélectionnez le fichier XAML (par exemple, **MyButton.xaml**) et fermez la boîte de dialogue
5. Sélectionnez la propriété **Configuration**.

## CONFIGUREZ LE BOUTON

La boîte de dialogue de configuration affiche un aperçu du fichier XAML sélectionné. Tous les éléments nommés dans le fichier XAML sont mentionnés dans l'arborescence :



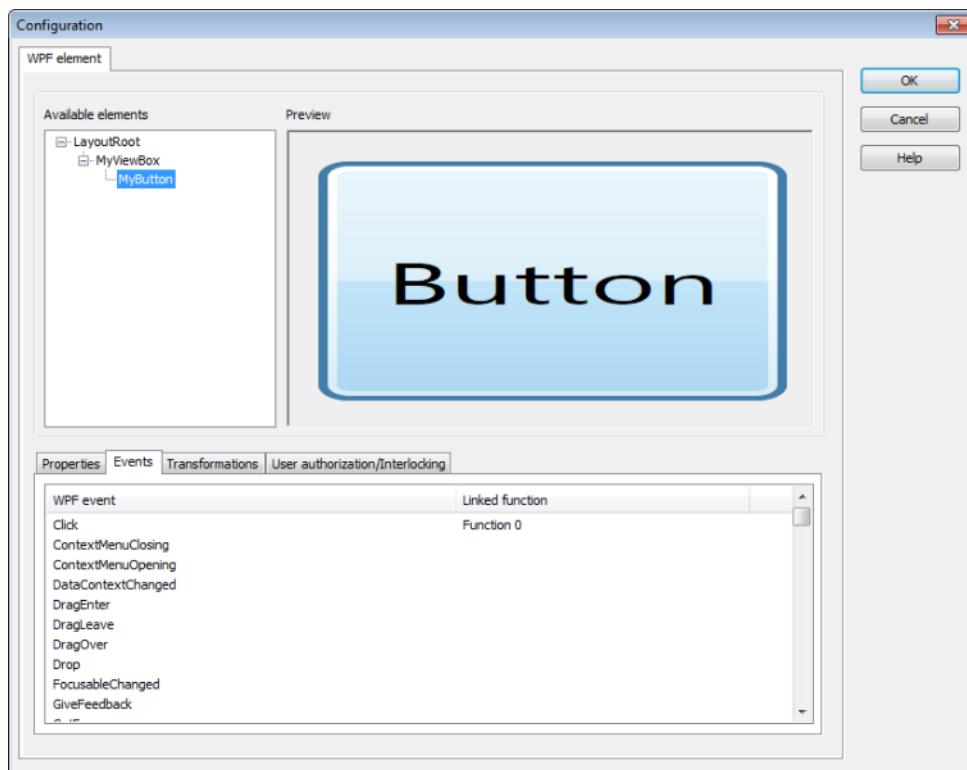
1. Sélectionnez le bouton WPF, qui se trouve dans **LayoutRoot->MyViewBox->MyButton**.
2. Consultez l'onglet **Content (Contenu)** de l'entrée **Propriétés** ; celui-ci contient le texte du bouton.
3. Cliquez sur la colonne **Link type** (Type de lien).

4. Sélectionnez **Variable** dans la liste déroulante.
5. Cliquez sur la colonne **Lien**.
6. La boîte de dialogue de sélection de variables s'affiche.
7. Sélectionnez la variable *int* pour lier cette variable à la propriété **Contenu**.

## ÉVÉNEMENTS

Pour attribuer des événements :

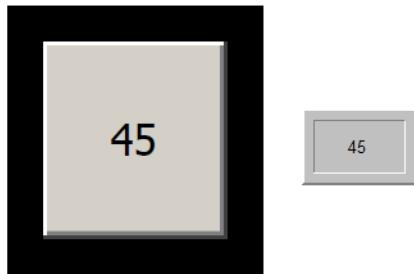
1. Sélectionnez l'onglet Événements.



2. Recherchez l'entrée "Clic" ; cet événement est déclenché par l'élément WPF dès que l'utilisateur clique sur le bouton.
3. Cliquez sur la colonne **Link type** (Type de lien).
4. Sélectionnez **Fonction** dans la liste déroulante.
5. Cliquez sur la colonne **Lien**.
6. La boîte de dialogue de sélection de fonctions s'ouvre.
7. Sélectionnez **Function\_0**.
8. Confirmez les modifications en cliquant sur **OK**.
9. Insérez un **élément valeur numérique** dans le synoptique.
10. Vous pouvez également lier cet **élément valeur numérique** aux variables *int*.

11. Compilez les fichiers du Runtime et démarrez le Runtime.

L'**élément WPF** est affiché dans le Runtime ; le texte du bouton est *0*. Lorsque vous cliquez sur le bouton, l'événement **Clic** est déclenché et la fonction **valeur prescrite** est exécutée. La valeur *45* est transmise directement au matériel, et la **valeur numérique** et le **bouton** affichent tous deux la valeur *45*.

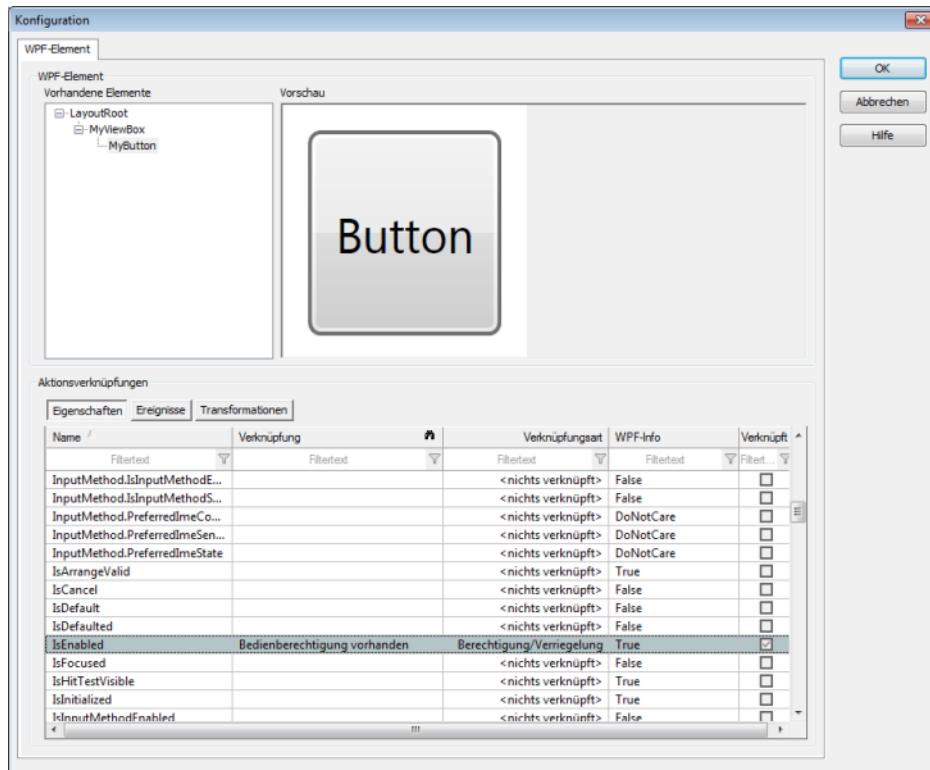


Définissez une valeur prescrite de *30* via l'**élément valeur numérique** ; cette valeur est alors reprise par l'**élément WPF**.

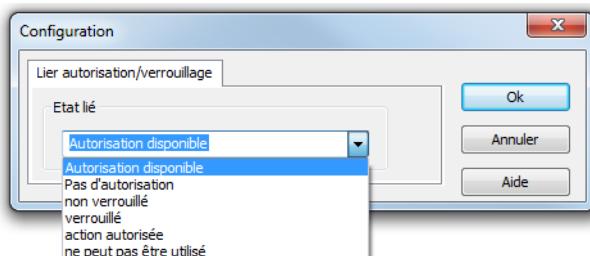
## AUTORISATION

A l'image d'une **valeur numérique**, un élément **WPF** peut être verrouillé en fonction des autorisations (symbole représentant un cadenas), ou être déverrouillé pour être actionnable. Réglez le niveau d'autorisation utilisateur sur *1* pour l'**élément WPF** et créez un utilisateur appelé **Test** possédant le **niveau d'autorisation 1**. Ensuite, définissez les fonctions **Connexion avec la boîte de dialogue** et **Déconnexion**. Liez ensuite ces deux fonctions aux deux nouveaux boutons avec texte affichés sur le synoptique.

Dans la boîte de dialogue de configuration de l'**élément WPF**, sélectionnez le bouton WPF **MyButton**, puis sélectionnez l'onglet **Propriétés**.

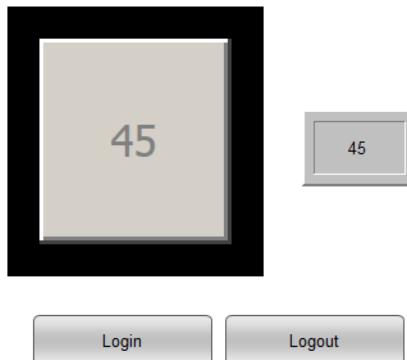


1. Sélectionnez l'élément **.IsEnabled**.
2. Cliquez sur la colonne **Link type** (Type de lien).
3. Sélectionnez **Authorizations/interlocking** (Autorisations/verrouillage) dans la liste déroulante.
4. Cliquez sur la colonne **Lien**.
5. Dans la liste déroulante, sélectionnez l'option *Authorized* (Autorisé).



6. Fermez la boîte de dialogue en cliquant sur **OK**.

Compilez le fichier Runtime ; notez bien que vous devez également sélectionner Authorizations to be Transferred (Autorisations utilisateur à transférer). Après le démarrage du Runtime, le bouton WPF apparaît désactivé sur le synoptique et ne peut pas être actionné. Connectez-vous maintenant en tant qu'utilisateur **Test** ; le bouton est activé et peut être actionné. Le bouton est verrouillé à nouveau dès que vous vous déconnectez.



## TRANSFORMATION

Les fichiers XAML doivent encore être adaptés pour que les transformations soient utilisables :

1. Basculez vers le programme **Expression Blend** .
2. Sélectionnez **MyButton**, afin que les propriétés de l'élément soient visibles dans la fenêtre des événements.



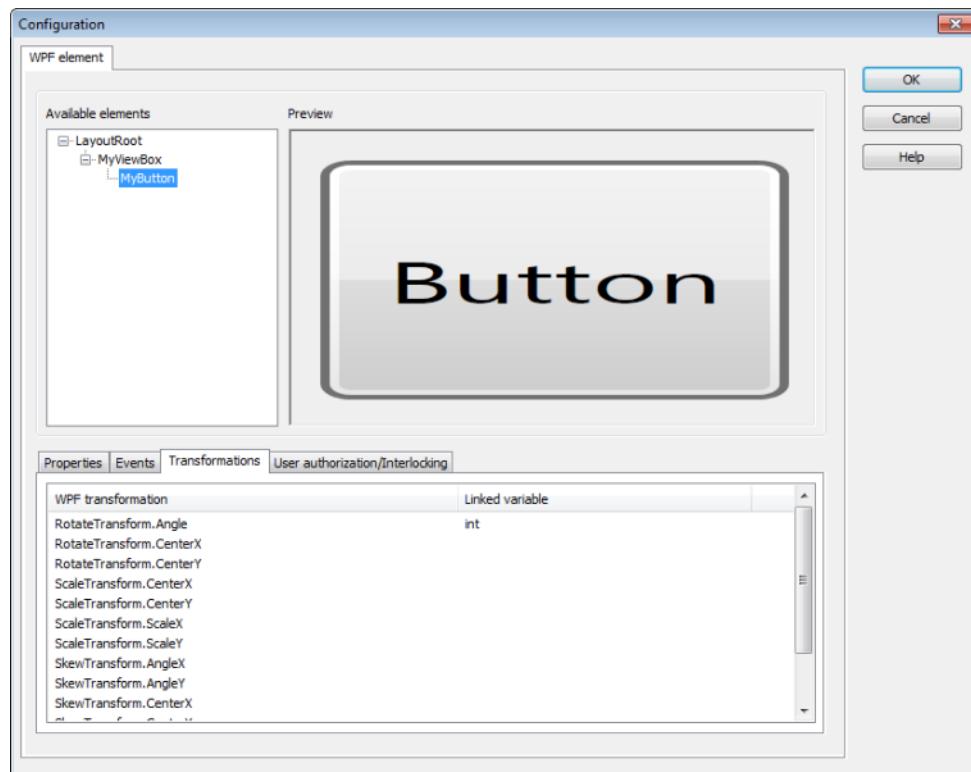
3. Sous **Transform (Transformer)** au niveau de l'entrée **RenderTransform (Transformation de rendu)**, sélectionnez l'option **Apply relative transform (Appliquer une transformation de rendu)**.

Un bloc est alors inséré dans le fichier XAML, dans lequel les paramètres de transformation dans le Runtime sont enregistrés.

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

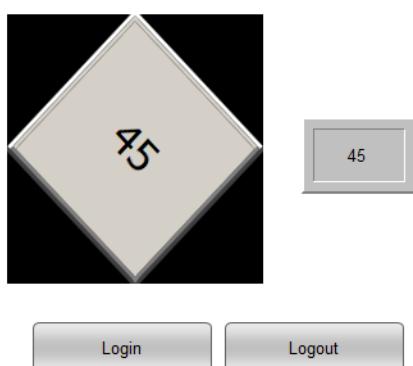
4. Enregistrez le fichier et remplacez l'ancienne version dans zenon par ce nouveau fichier.
5. Ouvrez à nouveau la boîte de dialogue de configuration de l'**élément WPF** :
  - a) Sélectionnez le bouton **MyButton**.

- b) Sélectionnez l'onglet **Transformations**.



- c) Sélectionnez l'élément **RotateTransform.Angle**.  
d) Cliquez sur la colonne **Link type** (Type de lien).  
e) Sélectionnez **Transformations** dans la liste déroulante.  
f) Cliquez sur la colonne **Lien**.  
g) La boîte de dialogue de sélection de variables s'affiche.  
h) Sélectionnez la variable **int** pour lier cette variable à la propriété **RotateTransform.Angle**.

Compilez les fichiers du Runtime et démarrez le Runtime. Connectez-vous en tant qu'utilisateur **Test**, puis cliquez sur le bouton. Le bouton comporte la valeur 45 et l'**élément WPF** pivote de 45°.



### 6.7.3 Integrate DataGrid Control in zenon

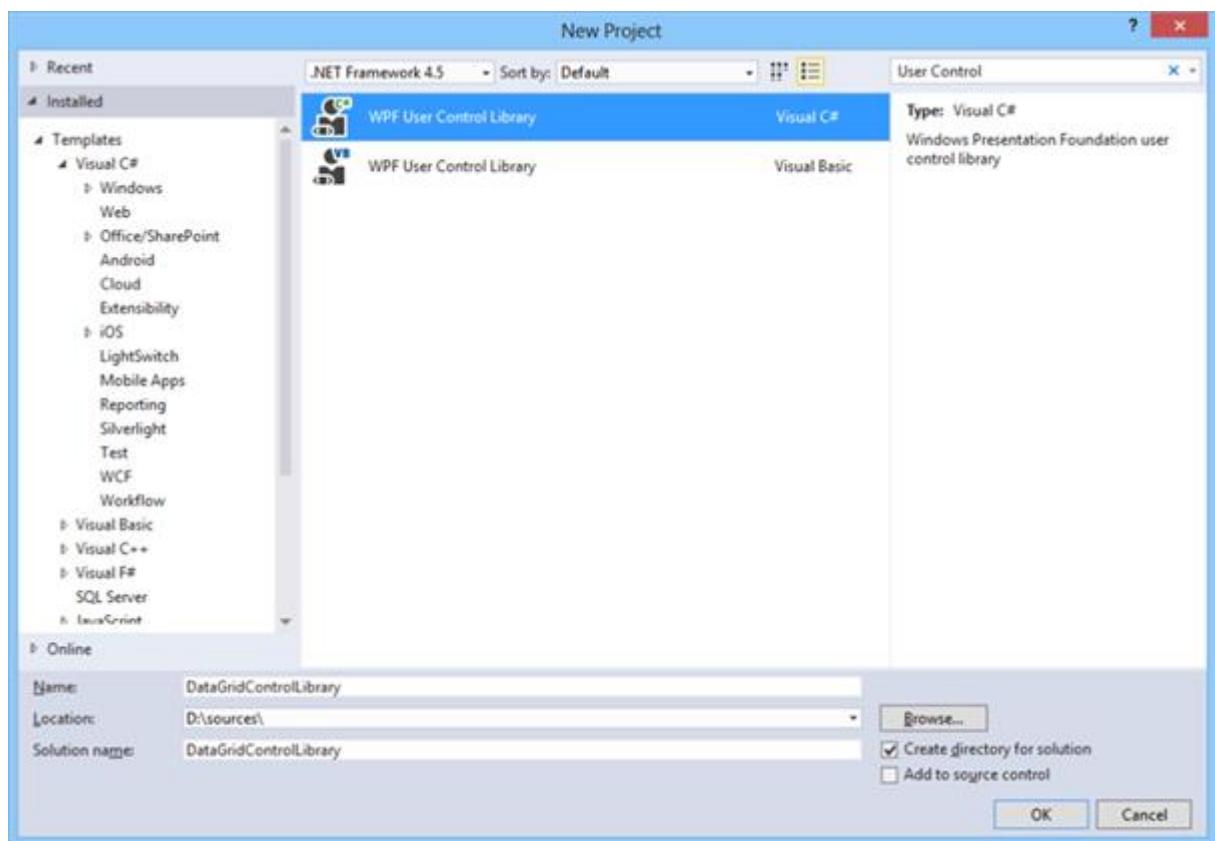
To create DataGrid control for zenon, you need:

- ▶ 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 4.6.2 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 **DataGridColumnLibrary**.

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.

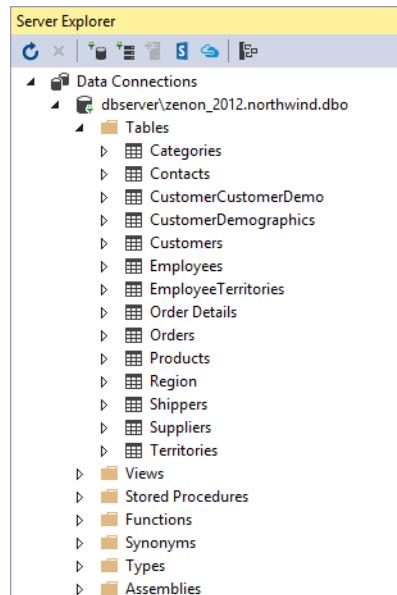
To set up the database connection:

- 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.

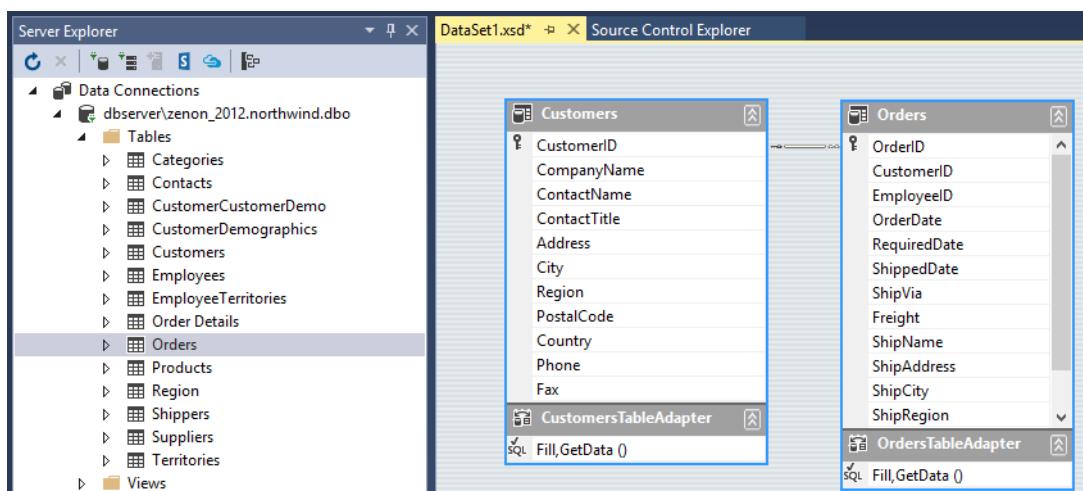
After adding the connection, the Server Explorer window should look a little like this:



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:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/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

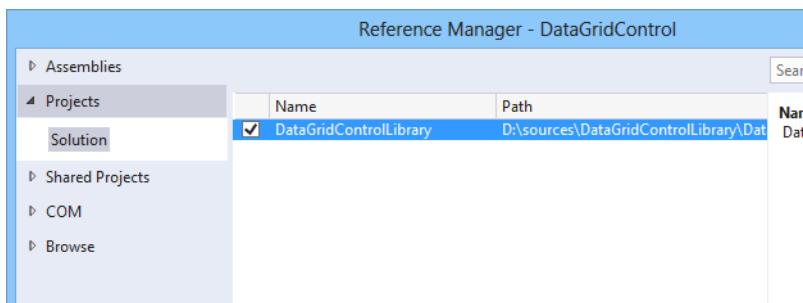
Now build the solution. The corresponding DLL (**DataGridColumnLibrary.dll**) is created in the output folder of the project.

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/expression/blend/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>
```

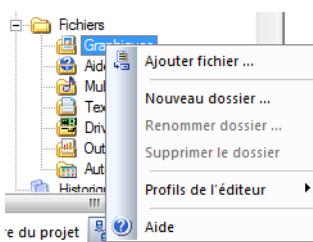
The line `xmlns:dataGridLibrary="clr-namespace:DataGridControlLibrary;assembly=DataGridControlLibrary"` defines the namespace **dataGridLibrary** and stipulates that this should use the assembly that has been created.

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.

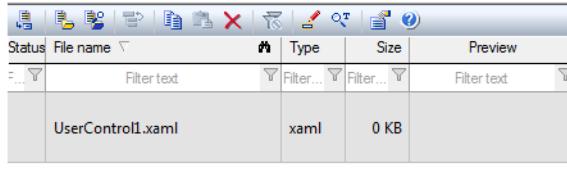
In the next step, how the DLL and XAML file are added to zenon is explained.

## STEPS IN ZENON

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



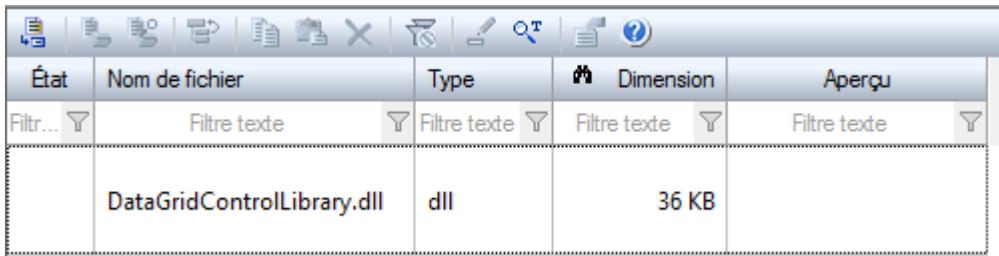
- Select the XAML file at the save location (**UserControl1.xaml** from the **DataGridControl** project) and add this:



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

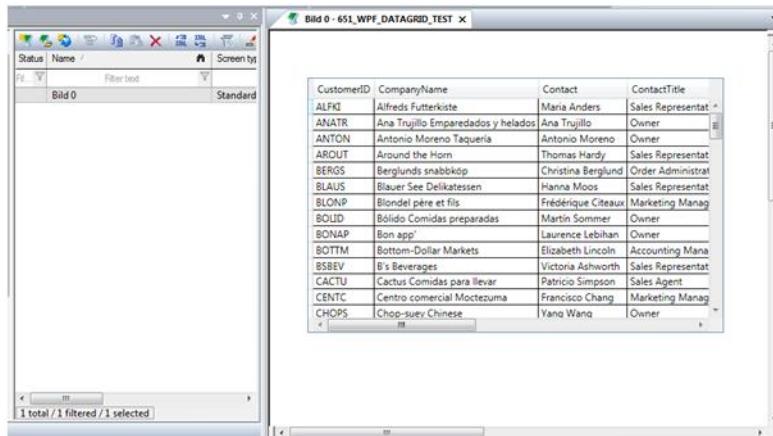
To do this:

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



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

You should now see the following in the zenon Editor:



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

## 👉 Conseil

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.