Tab Control Reference

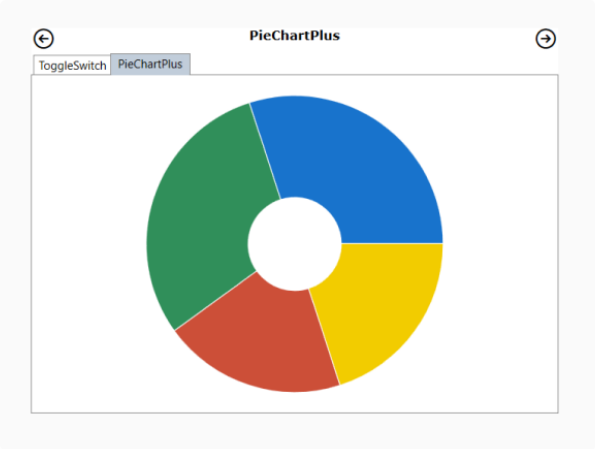
Organize content into tabbed pages.

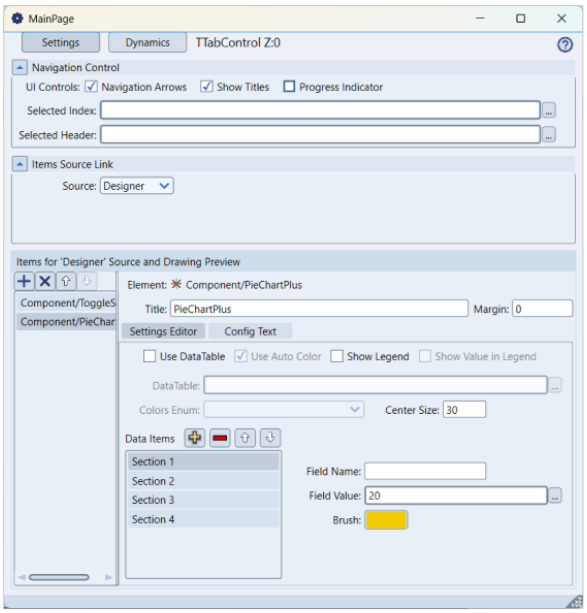
Reference Controls Viewer Tab Control Reference

The TabControl is a UI component designed to display a serie of items, such as ChildDisplays, Symbols, images, a tabbed selection menu. It allows users to navigate through items with configurable options for item selection and visual elements like titles and progress indicators. It supports item linking from external sources, setting limits on item count, and can be customized for auto cycling, navigation controls, and hover-based pauses.

The child elements and their properties can be defined in Designer mode for preview, as well as for dynamic queries from real-time tags, database queries, and other sources.

Requirements
Configuration
Runtime Execution





Requirements

This component is Portable. It runs both on Windows and on Web Pages hosted in any platform.

Configuration

To configure a Carousel:

1. Go to **Displays / Draw**.
2. On the Components Panel, select **Viewer**, then **TabControl**.
3. Click or drag-and-drop it on the Drawing area to use it.
4. Double-click the component to open the configuration window.

Carousel Settings	
Field	**Description**
Selected Index	Defines the index of the current active item in the TabControl. This value can be set manually or updated automatically when navigating through items.
Selected Header	Specifies the title of the currently active item in the TabControl. It updates based on the selected index and can be set manually.

<table>
<tr><td>Source</td><td>Determines the data source for the items in the TabControl. Options include Designer for static input, StringTag for tagged text elements, ArrayTag for data lists, and DataTable for structured data types.

Designer: Enables manual configuration of items through the Carousel Dialog interface. This method is ideal for straightforward setups involving fixed items that do not require frequent dynamic updates.

StringTag: Links item properties to dynamic SCADA tags. This allows the configuration of items that respond to real-time data changes. Used for dynamically updating symbols based on Tag values.
You can configure such tag with the following syntax:
"#Element = {Component found in T.Portable Library} #Property1 = {Property 1 Value} #Property2 = {Property 2 Value}".
An example would be:

StringTag Example
<pre>@Tag.tagText = "#Element=T.Portable.Controls.TCenterValueCircularGauge #LinkedValue=Tag.tag2;
#Element=T.Wpf.RunControls.TAssetsTree #LinkedValue=Tag.tag2;";</pre>
ArrayTag: Uses a string array where each entry defines an item and its properties. This enables dynamic generation of multiple items at once, useful for scenarios with several items. Each item follows the same format as StringTag.
The format used for this item is the same as StringTag, however you'll be using an Array for easier formatting.
Example:

ArrayTag Example
<pre>@Tag.textArray[0] = "#Element=T.Portable.Controls.TSemiCircleCircularGauge #LinkedValue=Tag.
tag2;";
@Tag.textArray[1] = "#Element=T.Wpf.RunControls.TDataGrid #LinkedValue=Tag.textArray;";</pre>
DataTable: Uses a DataTable or UserType for advanced configurations. Each row in the table represents an item, and each column defines the item's properties. Ideal for configurations that source data from databases or templates.
The format used here is similar, however, both the Element Name and its properties are now column names, and each row will be responsible for one item. An example would be the following:

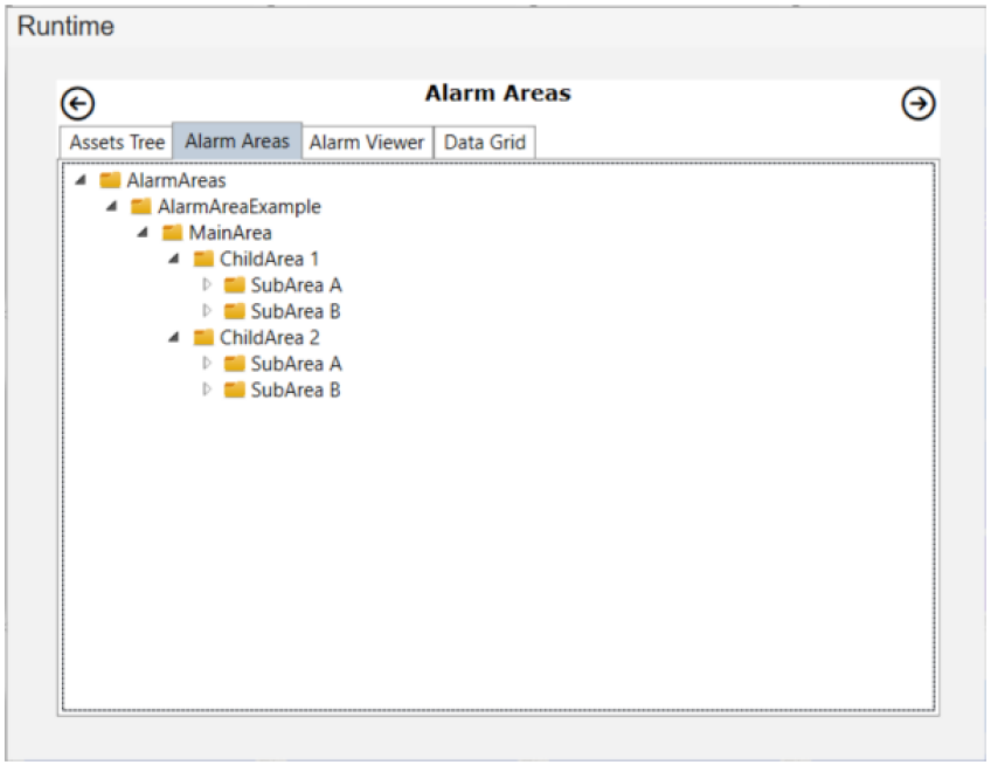
<table><tr><th>element</th><th>color</th><th>state</th></tr><tr><td>LampSymbol</td><td>red</td><td>on</td></tr><tr><td>MotorSymbol</td><td>green</td><td>off</td></tr></table>
Alternatively, you can also use a Tag DataTable with a UserType as Parameter, in which case your DataTable should have the following format:

<table><tr><th>member</th><th>index 0</th><th>index 1</th></tr><tr><td>LampSymbol</td><td>red</td><td>on</td></tr><tr><td>MotorSymbol</td><td>green</td><td>off</td></tr></table></td></tr>
<tr><td>Max Items</td><td>Limits the maximum number of items displayed in the TabControl. This value can be set to restrict the number of items loaded from the source.</td></tr>
<tr><td>Reload</td><td>Updates the TabControl menu to reflect any changes made to the source or settings. Clicking this button refreshes the TabControl content.</td></tr>
<tr><td>Navigation Arrows</td><td>Toggles the visibility of navigation arrows in the TabControl, allowing users to manually switch between items.</td></tr>
<tr><td>Show Titles</td><td>Enables the display of item titles in the TabControl, making it easier to identify each item in the TabControl.</td></tr>
<tr><td>Progress Indicator</td><td>Shows a visual indicator of the current position within the TabControl, helping users see how many items there are and which one is active.</td></tr>
<tr><td>Element</td><td>Represents an individual text item in the TabControl. The element can have specific configurations like font size, margins, and linked values.</td></tr>
<tr><td>Settings</td><td>Contains configuration details for individual items, such as linked values, font size, and margin settings, to define the visual layout of items.</td></tr>
</table>

Runtime Execution

The TabControl component allows displaying multiple tabbed items, where each tab can contain distinct content. It provides a structured interface for organizing and switching between different views or sections within the same space.

The TabControl automatically updates the Selected Index and Selected Header links to reflect the currently visible tab. These links can be used for dynamic interaction with other components, such as updating values or triggering logic based on the selected tab.



In this section...
