TextBlock Control Reference

Display static text content.

Reference **Controls** **Interaction** TextBlock

The TextBlock is a graphical element that displays textual information on screens.

Requirements
Configuration
Runtime Execution

Requirements

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

Configuration

After dragging and dropping the component into the Drawing area, double-click it to open the graphical element configuration window.

TextBlock Settings	
Field	**Description**
Text {withTags}	Specifies the text displayed in the TextBlock, including any linked tags. For instance: `Temperature: {@Tag.Temperature.Value}` In this example, it enables dynamically displaying the temperature value from the specified tag. The @ prefix MUST appear inside the curly braces — a bare `{Tag.X}` without @ renders the literal text instead of interpolating.
Format	This option is available only when you link an object to the button, and allows to define an expression to format the selected label. For example: N1 (number with 1 decimal place).
Text Wrapping	When checked, enables text wrapping within the TextBlock. This ensures that the text will flow onto the next line if it exceeds the width of the TextBlock.
Text Alignment	Horizontal alignment of the text inside the TextBlock box: **Left** (default), **Center**, **Right** or **Justify**. Set it with the **Align** box in the Text section of the Properties Sidebar. The alignment renders the same way in the desktop and the web client. Available from FrameworX 10.1.6, in the Designer and as the `TextAlignment` property when you build displays in JSON or through the AI tools. Displays without the property keep Left alignment.
Text Preview	This option is available only when you link an object to the button. Preview options control the preview of the label content or data bindings. These options indicate the data linked to the button. **Linked tag**: Displays the object path of a linked tag, such as 'Tag.Temperature.Value'. **Whitespace**: Shows a white space as a placeholder. **Placeholder**: Displays the symbol '#' as a placeholder.

Only the text inside the curly brackets, {}, will be dynamic.

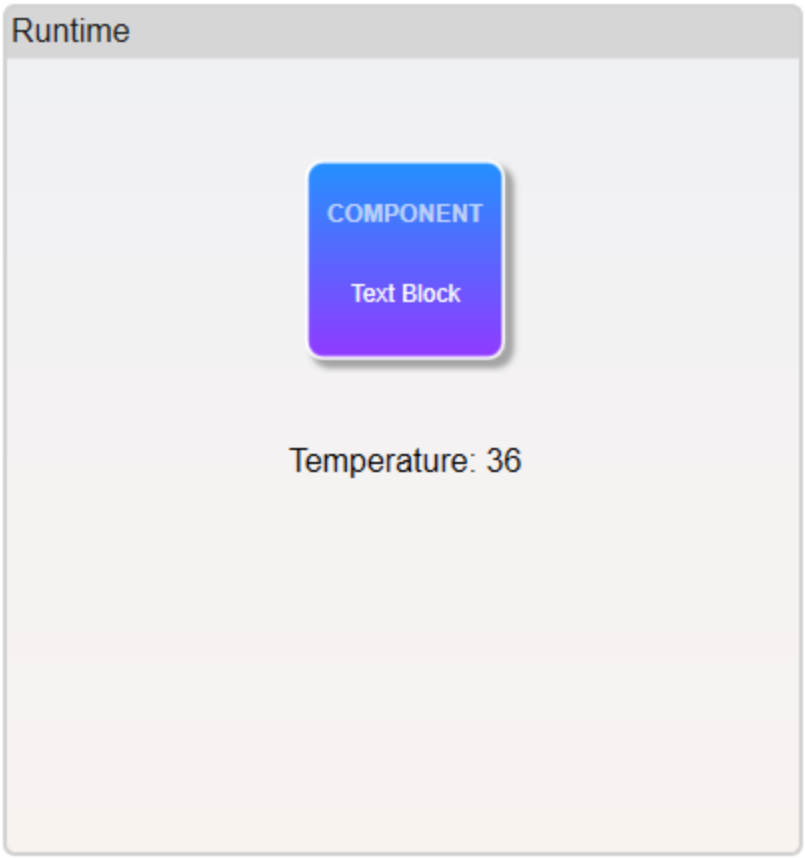
For example, if you want to use symbol labels, like #Text:abc, you need to write it as: {#label1:abc}

The evaluation of the expression: "Text1 {#label1:abc} text2" is "Text1 abc Text2", or abc being replaced by the symbol configuration.

JSON authoring (DesignerMCP / ConsoleMCP / write_objects): for any runtime tag binding, emit the `LinkedValue` property — not the `Text` property. `Text` is design-time fallback only and does NOT interpolate `{...}` or live-bind `@Tag.X`. `LinkedValue` accepts both the bare-reference form `"@Tag.X"` (renders just the value) and the composite-template form `"Temperature: {@Tag.Reactor/Temp} °C"` (literal text + curly-brace interpolation; @ required inside braces).

Runtime Execution

When the application needs to display information, it uses a text block to show read-only text. The TextBlock presents a static text that can display dynamic information through expressions. Users can view but not modify this information during runtime execution.



In this section...
