Button Control Reference

Trigger actions with a clickable button.

Reference **Controls** **Interaction** Button

The Button Control in the software platform allows users to create interactive buttons for applications. It includes a mechanism for initiating actions or commands within the interface. The functionality includes customizable label text with positioning options.

Requirements
Button vs PushButton (or ToggleButton)
Configuration
    Label Settings
    Hover Color
    Dynamics
Runtime Execution

Requirements

The TButton is a Portable component, functioning on both Windows and web pages across any platform. It is compatible with WPF and HTML5 displays, offering versatile use without any specific requirements.

Button vs PushButton (or ToggleButton)

The Button and PushButton controls have similar appearance, but there are distinctions on how typically they are used.

The Button is typically used in combination with the Action Dynamic. When there is an action connected with the Button, its appearance will automatically act to the Mouse Over events, and when is pressed, a visual feedback is provided. The pressed visual stays only while the button is pressed, there is no state or dynamic property connected with the Button.

The PushButton (or ToggleButton), on the other hand, can be in either a Checked or Not Checked state, with its appearance reflecting the current state.

See PushButton Control Reference for more information.

Configuration

Button configuration include Label Text for button naming, Localize checkbox for localization, and Label Settings for positioning (Auto, Top, Left, Inside, None). Width settings offer manual adjustment or an Auto checkbox set to 80.

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

Button Settings	
Field	**Description**
Label Text	Specifies the text displayed in the button.
Localize	Enables the label text to be dynamically adjusted based on the user's locale settings, ensuring the interface is accessible and understandable for users from different language backgrounds.

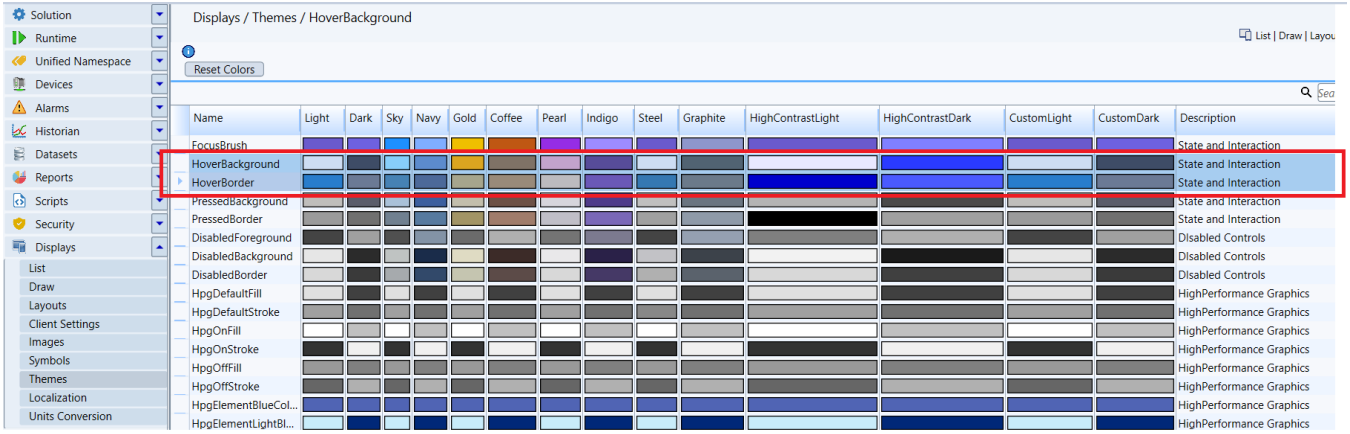
Label Settings

Label Settings	
Field	Description

Position	Defines the label's position. There are the following options: **Auto**: The label position is determined automatically based on the available space and button size. **Top**: The label is positioned above the button. **Left**: The label is positioned to the left of the button. **Inside**: The label is positioned inside the button, usually centered. **None**: The button has no visible label.
Width	Defines the label's width.
Auto	When enabled, defines the label's width automatically.
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.
Format	This option is available only when an object is linked to the button. It allows you to define an expression to format the selected label. For example, 'N1' formats a number with one decimal place.

Hover Color

The colors of the button hover effect can be redefined under Display > Themes by adjusting the HoverBackground and HoverBorder options for the active theme.



JSON authoring — LabelLink, not Label Text

When authoring Buttons via DesignerMCP / ConsoleMCP / `write_objects`, the underlying JSON property name is `LabelLink` (not `Label Text`, which is the Designer-UI display name). Accepted forms per the four-family rule:

- **Static text:** `"LabelLink": "Start Motor"`
- **Tag binding:** `"LabelLink": "@Tag.MotorState"`
- **Composite template:** `"LabelLink": "Setpoint: {@Tag.Reactor/SP} °C"` — the @ prefix is required INSIDE the curly braces.

For SetValue actions on a Button: the action lives in the `Dynamics` array as `ActionDynamic` with `MouseLeftButtonDown ActionType : "SetValue" ObjectLink` (target) + `ObjectValueLink` (the value, per the four-family rule on Dynamic Behavior Control Reference). String literals must be escape-wrapped: `"ObjectValueLink": "\"Dark\""` — a bare `"Dark"` silent no-ops.

Dynamics

Usually, configuring Graphic Elements also involves defining Dynamics for them. A dynamic defines how graphic elements respond to user inputs and other events, including handling clicks, updating displays in real-time, and dynamic data visualization. Refer to Dynamic Behavior Control Reference for more information.

Runtime Execution

When users click a button, the application executes the associated command or action immediately. The button provides visual feedback, such as changing color or appearance, to indicate it has been pressed. This triggers event handlers in the solution, which carry out the designated operation, such as starting a machine, submitting data, or opening a new interface.



In this section...
