# Add Symbols to Displays Tutorials

Add and customize Symbols on Displays — create reusable visual components, expose configurable Labels, build a Symbol Library, and place Symbols on Displays bound to Tags.



## This Tutorial Teaches you to:

- Create custom Symbols
- Add Symbol properties (Labels)
- Build Symbol libraries
- Apply Symbols to Displays

This Tutorial Teaches you to:
1. Create a Basic Symbol
2. Add Symbol Properties (Labels)
3. Build a Symbol Library
4. Apply Symbols to Displays
5. Verify at Runtime

### Prerequisites

- Design Canvas Displays (Tutorial)
- Design Dashboard Displays Tutorials

---

## 1. Create a Basic Symbol

1. Navigate to **Displays  Symbols**.
2. Create a new Symbol named `Motor`.
3. Draw the motor graphic on the design surface:
    - Circle for the body
    - Rectangle for the base
    - "M" text label
4. Save the Symbol. It is now available under your custom Symbol category in the Symbol Library.

Browse the built-in Symbol Library first via the **Symbols** tab — most operator-interface elements (HMI gauges, equipment shapes such as TANK / VALVE / PUMP / MOTOR / BLOWER, HPG flat themed primitives) already exist. Define a new Symbol only when no library Symbol fits.

---

## 2. Add Symbol Properties (Labels)

Labels are the parameters a Symbol exposes to the Display that places it — they are what makes a Symbol reusable across many Tags. Define Labels inside the Symbol; bind them to Tags at the placement site.

1. Open the Symbol in **Displays  Symbols**.
2. Select an element inside the Symbol (for example the body Circle).
3. In a text property such as **LinkedValue** or a Dynamic binding, use the `#<LabelName>:<OptionalDefault>` syntax to declare a Label. Example: `#State:0` on a FillColor Dynamic declares a **State** Label with default `0`.
4. Inside the Symbol definition, reference the Label as `@Label.State` in expressions or Dynamics. The Label list is auto-extracted from the `#` markers on the Symbol's elements and surfaces in the **Labels** column of the **DisplaysSymbols** table.

Common platform Labels used by built-in Symbols: **State** (on/off), **Rotate** (rotation toggle), **RPM** (rotation speed), **Level** / **Max** / **Min** (level gauge), **Value**, **Title**. Wizard Symbols (TANK, VALVE, PUMP, MOTOR, BLOWER) carry a fixed Label set per category.

> **Use-site vs definition-site rule.** `@Label.<Name>` is valid ONLY inside a Symbol definition (a row of **DisplaysSymbols**). When placing a Symbol on a Display (a row of **DisplaysList**), bind Labels to Tags through the placement's **SymbolLabels** property using `@Tag.` paths — never `@Label.`. Mixing the two is the single most common Symbol-binding mistake; `@Label.` on a Display element silently fails to resolve at runtime.

---

## 3. Build a Symbol Library

A Symbol Library is the collection of reusable Symbols available to every Display in the Solution. Each row of **DisplaysSymbols** is one Symbol in the Library. Two organizational levers:

- **Name with a path prefix** — Symbol names accept `/`-separated folders (e.g., `Motors/HighSpeed/Motor3Phase`, `HMI/Gauges/Tacho`). The Symbols panel renders the folders as a tree.
- **Category** — set the **Category** field to group Symbols across folders for filtering in the Symbols panel.

Modifying an existing Symbol affects every Display that places it. To update the Library:

1. Right-click the Symbol instance on a Display and choose **Edit Symbol <Name>**.
2. Make the change on the Symbol design surface.
3. Right-click and choose one of:
    - **Save to Symbol Library** — updates the Symbol globally; every Display that places this Symbol picks up the change.
    - **Save Only to this Display** — converts this placement into a local copy; the Library Symbol is unchanged.

> **Save to Symbol Library is a global update.** The change applies to every Display in the Solution that places this Symbol — including Displays you have not opened. If you only want to change this one placement, choose **Save Only to this Display** instead.

---

## 4. Apply Symbols to Displays

Placing a Symbol on a Display is the common operation (most tasks). The placement is a row of **DisplaysList** elements with `Type="Symbol"` and a **SymbolLabels** array that binds the Symbol's Labels to `@Tag.` paths.

1. Open the target Display.
2. From the Symbols panel, drag the Symbol onto the canvas (Canvas Display) or into a grid cell (Dashboard Display).
3. With the placed Symbol selected, open **SymbolLabels** in the Properties pane.
4. For each Label, set **Value** to an `@Tag.` path or a literal. Example bindings for a `Motor` Symbol with State / RPM / Title Labels:
    - `State @Tag.Plant/Line1/Motor1/Running`
    - `RPM @Tag.Plant/Line1/Motor1/Speed`
    - `Title "Line 1 Motor"` (static text)
5. Resize the placement to fit the layout — Symbols are vector-based and auto-scale, so the placement Width/Height do not need to match the Symbol's design size.

The same Symbol can be placed many times on the same Display or across many Displays, each placement bound to a different set of Tags through its own **SymbolLabels**.

---

## 5. Verify at Runtime

1. Start the Runtime (RichClient or WebClient).
2. Open the Display containing the Symbol placement.
3. Drive the bound Tag values and confirm the Symbol reacts — the Label-bound Dynamics (color, rotation, text) update live.
4. Open the Display in the other client family (WebClient if you tested in RichClient, and vice versa) to confirm cross-platform behavior is consistent.

---

### In this section

---