Displays Symbols Reference

Reuse graphical symbols in displays.

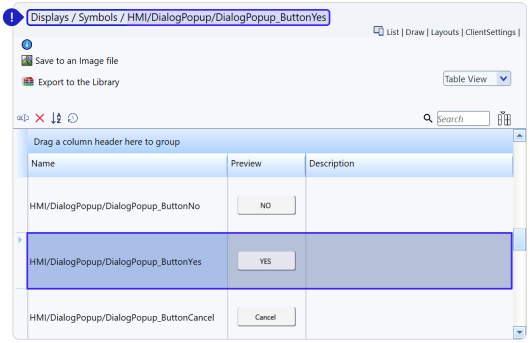
Displays Symbols (Reference) are vector-based graphical elements representing engineering components with dynamic properties that reflect real-time system states through data point mappings.

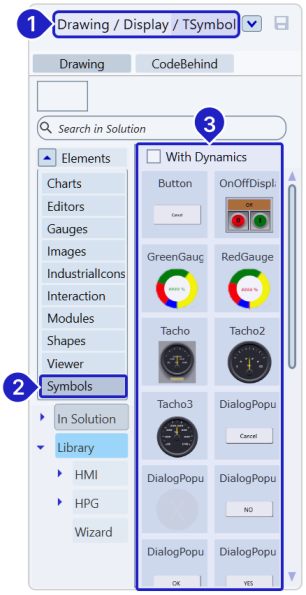
Display Symbols provide:

- Vector-based scalable graphics
- Dynamic property binding
- Reusable symbol libraries
- Real-time state visualization
- Theme-aware coloring

Symbols serve as visual representations of devices, processes, and functions, dynamically updating based on connected data points.

Symbols Libraries And Collections
Symbol Management
Symbol Properties
Dynamic Properties
Using Symbols
Symbol Organization
Symbol Editor Features
Performance Optimization
Import/Export
Best Practices
Troubleshooting
Advanced Features





Symbols Libraries And Collections

Collections comprise graphical elements for software interfaces. The Local Symbols Library allows symbol creation and management. The Symbol Factory Library offers a range of symbols for use. HMI Collections provide symbols for Human-Machine Interfaces, and HPG Collections deliver graphics for advanced visualizations. These resources support interface development with dynamic, data-responsive features.

The Symbols Library is composed of nearly 2000 Symbols organized in the following collections.

Local Solution

User-defined and default symbols specific to the solution:

- Custom dynamic properties
- Tag mappings
- Solution-wide updates
- Reusable components

HMI Collection

Industrial interface elements organized by equipment type:

- **Categories**: Blowers, Conveyors, Pumps, Valves
- **Subcategories**: Specific equipment variants

- **States**: Multiple visual states per symbol
- **Controls**: Interactive elements

HPG Collection

High Performance Graphics per ISA 101 standard:

- Optimized for operator effectiveness
- Minimal cognitive load
- Clear state indication
- Consistent visual hierarchy

For more information on High-Performance HMI Graphics, see High Performance HMI

Symbol Management

Access Points

- **Displays Symbols** - Create, edit, manage library
- **Displays Draw** - Insert symbols into displays
- **Symbol Library Panel** - Browse and select

Creating Symbols

1. Navigate to **Displays Symbols**
2. Click **New Symbol**
3. Design using drawing tools
4. Add dynamic properties
5. Save to library

Note: A symbol cannot contain another symbol. Nested symbols are not supported, and the Designer blocks inserting a symbol into the symbol currently being edited.

Symbol Properties

Property	Description	Type
ID	Unique identifier	Auto
Name	Symbol name	String
Preview	Visual representation	Image
Version	Release number	Integer
Size	Dimensions (width x height)	Point
HasDynamics	Contains dynamic elements	Boolean
Category	Organization group	String
EditSecurity	Edit permissions	Security
LockState	Edit lock status	Boolean
Description	Purpose and usage	String

Dynamic Properties

Binding Data

Map symbol properties to tags:

xml

```xml
<Symbol Name="Motor">
  <Fill>{Tag.Motor1.Running ? "StateGreen" : "StateRed"}</Fill>
  <Rotation>{Tag.Motor1.Speed / 100 * 360}</Rotation>
  <Visibility>{Tag.Motor1.Visible}</Visibility>
</Symbol>
```

Common Dynamics

- **Color Change** - State indication
- **Rotation** - Speed/position
- **Visibility** - Show/hide
- **Size** - Scaling
- **Position** - Movement
- **Text** - Value display

Using Symbols

Inserting into Displays

1. Open display in Draw mode
2. Select symbol from library
3. Drag to display canvas
4. Configure properties
5. Map to data points

Symbol Instances

Each instance maintains:

- Independent property values
- Tag mappings
- Runtime state
- Event handlers

Symbol Organization

Categories Structure

```
HMI Collection
   Process Equipment
    ? Pumps
    ? Valves
    ? Motors
   Instrumentation
    ? Gauges
    ? Indicators
   Containers
    ? Tanks
    ? Vessels
```

Naming Conventions

- **Prefix**: Category (MOT_, VLV_, PMP_)
- **Base**: Equipment type
- **Suffix**: Variant (_H, _V, _01)

Symbol Editor Features

Drawing Tools

- Vector shapes
- Path editing
- Gradient fills
- Theme colors
- Layer management

Dynamic Elements

- Expression editor
- Property binding
- Animation preview
- State simulation

Performance Optimization

Symbol Complexity

- Limit nested groups
- Optimize path points
- Reduce gradients
- Cache static elements

Dynamic Updates

- Minimize binding expressions
- Use efficient triggers
- Batch property changes
- Implement update throttling

Import/Export

Export Options

- **To Library** - Share across solutions
- **As Image** - PNG, SVG formats
- **To File** - Symbol package

Import Sources

- Symbol packages
- SVG files
- Legacy symbols
- Third-party libraries

Best Practices

- **Use Vector Graphics** - Ensure scalability
- **Apply Theme Colors** - Support light/dark modes
- **Minimize Complexity** - Optimize performance
- **Create Variants** - Different detail levels
- **Document Symbols** - Clear descriptions
- **Test Dynamics** - Verify all states
- **Organize Libraries** - Logical categories

Troubleshooting

Symbol not updating:

- Check tag connections
- Verify expression syntax
- Review binding mode
- Test in runtime

Performance issues:

- Reduce symbol complexity
- Optimize expressions
- Check update frequency

Display problems:

- Verify theme colors
- Check scaling settings
- Review layer order
- Test different resolutions

Advanced Features

Custom Properties

csharp

```
// Add custom property to symbol
Symbol.AddProperty("AlarmState", typeof(int));
Symbol.AddProperty("MaintenanceHours", typeof(double));
```

In this section...
