Skill Display Construction - HMI Symbol Choice

Choosing the right symbol for every equipment family on a Canvas display — Solution first, then HPG, HMI, Wizard, and only primitives as a last resort.

Purpose

This page is the canonical reference for picking the right symbol when authoring a Canvas display. It expands **Skill Display Construction - Canvas** §4.5 (Choosing a symbol source) into a per-equipment-class lookup with sizing tables and the SymbolLabels keys each family exposes.

Load this skill when:

- The display you are building shows physical equipment — vessels, pumps, motors, valves, blowers, instruments, heat exchangers, conveyors.
- You are about to compose a piece of equipment from `Rectangle`, `Ellipse`, `Polygon`, or `Path` primitives. **Stop. Read this page first.** Primitives are the last resort, not the first move.
- You inherited a solution and want to keep new displays visually consistent with what is already there.

The cost of choosing the wrong source: a display that visually disagrees with the rest of the plant's screens, drifts under theme switches, or composes a vessel that is read as a sketch instead of as standard equipment. The cost of choosing the right source: usually one Symbol element with two SymbolLabels.

Section 1 — The symbol source order

Before composing any equipment from primitives, look for an existing symbol. The search order is fixed:

1. **Solution symbols.** `list_elements('Solution')` shows what the plant has already standardized on. If a `Solution/Tanks/Tank_Vertical` exists, use it — other displays in this solution already do. Mixing Solution symbols with Library symbols or Wizards for the same equipment type produces a screen set that visually disagrees with itself.
2. **HPG Library** (High Performance Graphics). `list_elements('Library/HPG')`. Flat, theme-aware symbols that respond to the standard state convention (`0`=Off/Stopped/Closed, `1`=On/Running/Open, `2`=Disabled/Out-of-Service) via the `HPOffFill`/`HPOnFill`/`HPDisableFill` theme brushes. This is the right default for operator control screens.
3. **HMI Library** for detailed/realistic symbols. `list_elements('Library/HMI')`. Traditional detailed symbols with shading and depth — right for training material, technical diagrams, and mechanical documentation. Wrong for live operator screens (too much visual noise competes with state communication).
4. **Wizard symbols** for fast-path canonical equipment. Five pre-wired Wizards (`BLOWER`, `MOTOR`, `PUMP`, `TANK`, `VALVE`) cover the most common cases — state and rotation dynamics are already wired internally.
5. **Compose from primitives** (`ShapeGroup` of `Rectangle` / `Ellipse` / `Path`) only when none of the above fits. This is the slowest path and the one most likely to drift across displays. Use the ShapeGroup recipe in Canvas §5 (Recipe 1 — Vessel with jacket).

Rule: when writing to an existing solution, run `list_elements('Solution')` before reaching for a Library symbol or a Wizard. Match what is already there.

Section 2 — Per-equipment-class recommended path

For each equipment family the operator is likely to name, this table maps to the recommended `SymbolName` path. **Always confirm the actual symbol exists in the current release with `list_elements(<parent path>)`** before referencing it — library content evolves between releases.

Equipment family	Operator vocabulary	Fast path	Detailed alternative	Composite fallback
Storage tank / Drum / Surge vessel	tank, drum, surge tank, day tank, silo	`Wizard/TANK`	`Library/HMI/Vessels/*`	`Cylinder` auto-shape (Canvas §3) or ShapeGroup vessel recipe (Canvas §5 Recipe 1)
Reactor / Mixer / Agitated vessel	reactor, mixer, agitator, CSTR	`Wizard/TANK` + impeller (Canvas Recipe 1 impeller block)	`Library/HMI/Vessels/*` + `RotateDynamic` on a child Rectangle	ShapeGroup vessel + impeller (Canvas §5 Recipe 1)
Pump (centrifugal / positive-displacement / dosing)	pump, centrifugal pump, dosing pump, metering pump	`Wizard/PUMP`	`Library/HMI/Pumps/*`	ShapeGroup composed from Ellipse + Rectangle + nozzle Polygons
Motor / Drive (decoupled from pump or fan)	motor, electric motor, drive, VFD output	`Wizard/MOTOR`	`Library/HMI/Motors/*`	Rectangle with `RotateDynamic` impeller (Canvas Recipe 1 impeller pattern)
Blower / Fan / Forced-draft unit	blower, fan, FD fan, ID fan, exhaust fan	`Wizard/BLOWER`	`Library/HMI/Blowers/*` or `Library/HMI/Fans/*`	ShapeGroup composed from Trapezoid + radial-arm Polygons
Valve (manual / motorized / on-off / modulating)	valve, gate valve, ball valve, control valve, butterfly valve	`Wizard/VALVE`	`Library/HMI/Valves/*`	ShapeGroup composed from two Polygon triangles meeting at center (the P&ID convention)

Heat exchanger / Cooler / Condenser	heat exchanger, shell-and-tube, plate heat exchanger, condenser, cooler	(no Wizard) — jump to detailed alternative	`Library/HMI/HeatExchangers/*`	Recipe 4 (reactor with heating coils, Canvas §5) or Path-overlay zigzag
Conveyor / Belt / Screw	conveyor, belt, screw conveyor, drag chain	(no Wizard) — jump to detailed alternative	`Library/HMI/Conveyors/*`	ShapeGroup composed from long Rectangle + Ellipse end rollers
Instrument / Sensor / Transmitter	sensor, transmitter, gauge, indicator, instrument bubble	(no Wizard)	`Library/HMI/Instruments/*` — ISA-5.1 / ISA-101 instrument bubbles	Ellipse with composite `LinkedValue` TextBlock label inside
Pipe / Line / Connector	pipe, line, header, connector, jumper	Not a symbol — use the `Gridline` primitive (Canvas Recipe 3)	—	—
Process compute / PLC / Controller / Skid	PLC, RTU, controller, skid, edge gateway, MQTT broker	(no Wizard)	`Library/HMI/Computers/*` or `Library/HMI/Controllers/*`	Rectangle + label TextBlock; use `Cloud` auto-shape for MQTT broker / cloud service

If `list_elements()` on the detailed-alternative path returns nothing matching the family, walk through the path one level higher (`list_elements('Library/HMI')`) and scan the subfolder names. The HMI library is ~1600 symbols deep — entries beyond what this table names exist for many narrow industries (pharma, oil & gas, water, food & bev, electrical, building automation).

Section 3 — Sizing per equipment family

All symbols scale to any proportional size. **Maintain aspect ratio — never stretch asymmetrically.** The base "recommended" sizes below are calibrated for a standard 1600×900 control-room Canvas; scale up proportionally on 1920×1080 or 4K-scaled canvases.

Equipment family	Minimum size	Compact	Standard	Hero (single equipment focus)
Tank / Drum / Vessel	60×90	80×120	120×180	200×320
Reactor / Mixer (vessel + impeller)	80×120	120×180	180×260	260×380
Pump	60×60	80×80	100×100	140×140
Motor / Drive	60×60	80×80	100×100	140×140
Blower / Fan	80×80	100×100	140×140	200×200
Valve	40×40	60×60	80×80	120×120
Heat exchanger	120×80	180×120	240×160	320×200
Conveyor (length × depth)	200×40	300×48	450×60	700×80
Instrument bubble	32×32	40×40	52×52	72×72

When in doubt, use Standard. The Compact column is for dense equipment grids (a 4×3 motor wall, a row of identical valves). Hero is for single-equipment focus displays (one reactor at the center of a process-area overview, one pump on a maintenance-detail screen).

Minimum sizes are hard floors: below them the symbol's internal labels and state indicators degrade past readable. If the layout pressures you below the minimum, redesign the layout — do not shrink the symbol.

Section 4 — SymbolLabels keys per family

`SymbolLabels` is the only way to push data into a symbol. `Key` matches a slot the symbol declares internally (case-sensitive). `LabelValue` is the binding (typically `@Tag.<path>`). `FieldType: "Expression"` is the default.

The keys exposed by the five Wizards (confirmed against Designer):

SymbolName	Key	Typical binding	Notes
`Wizard/BLOWER`	`State`	`@Tag.<blower>/Running`	`0`=Stopped, `1`=Running — gates the rotation dynamic and the state fill
`Wizard/BLOWER`	`RPM`	`@Tag.<blower>/Speed`	Drives the visible rotor speed
`Wizard/MOTOR`	`State`	`@Tag.<motor>/Running`	Same convention as BLOWER
`Wizard/MOTOR`	`RPM`	`@Tag.<motor>/Speed`	—
`Wizard/PUMP`	`State`	`@Tag.<pump>/Running`	—
`Wizard/PUMP`	`RPM`	`@Tag.<pump>/Speed`	Style variant (centrifugal vs PD) is a Designer-side configuration choice on the placed symbol, not a SymbolLabel
`Wizard/TANK`	`Level`	`@Tag.<tank>/Level`	0..100 (or whatever scale the symbol's internal fill is calibrated for)
`Wizard/TANK`	`Alarm`	`@Tag.<tank>/Alarm`	Boolean — triggers the alarm-state visual

`Wizard/VALVE`	`State`	`@Tag.<valve>/Open`	`0`=Closed, `1`=Open
`Wizard/VALVE`	`Position`	`@Tag.<valve>/Position`	0..100 for modulating valves; bind to the open-percent signal

Library/HPG and Library/HMI symbol keys vary by symbol. Call `list_elements('Library/HPG/<subpath>/<SymbolName>')` to discover the keys a specific symbol exposes. Common conventions:

- State-based symbols expose `State` as the primary boolean / enum.
- Level-based symbols (tanks, drums, hoppers) expose `Level` and often `HighAlarm` / `LowAlarm`.
- Flow-based symbols (pumps, blowers, valves) may expose `Flow` or `FlowRate` in addition to `State`.
- Instrument-bubble symbols (Library/HMI/Instruments) expose a `Tag` key for the ISA tag name shown in the bubble.

SymbolLabels rules — from Skill Display Construction - Basics §7:

- `Key` must match a label key defined inside the symbol (case-sensitive).
- `LabelValue` uses `@Tag.X` / `@Client.Context.X` / `@Now` / composite strings.
- **Never use `@Label.X`** in display-element SymbolLabels — `@Label.` is a symbol-definition internal only.
- Never bind via direct properties on the Symbol element itself — SymbolLabels is the only data path into a symbol.

Section 5 — Discovery tools

The runtime catalogues are the authoritative source — this page's tables are guidance, not a contract. Always discover before placing:

```
list_elements('Solution')              -- symbols already in this solution
list_elements('Library/HPG')           -- HPG library top level
list_elements('Library/HPG/Pumps')     -- HPG pumps subfolder
list_elements('Library/HMI')           -- HMI library top level (large, may truncate)
list_elements('Library/HMI/Valves')    -- specific HMI subfolder
list_elements('Wizard')                -- the 5 Wizard symbols
```

Truncation: `list_elements()` returns at most 50 entries per call. When the result includes `truncated: true`, drill into specific subfolders to see complete listings.

Section 6 — Common pitfalls

Mistake	Fix
Composing a vessel from primitives without checking for a Library or Wizard symbol	Run the source-order checklist in §1. Primitives are step 5, not step 1.
Mixing Solution / Library / Wizard for the same equipment type across one display set	Pick one source family per equipment class and stay there. Cross-source mixing is the #1 cause of "the displays don't look like they belong to the same plant."
Stretching a symbol asymmetrically	Maintain aspect ratio. Width and Height proportions must match the symbol's design ratio.
Setting a symbol's state by writing `Fill` or `Stroke` directly on the Symbol element	Direct properties are ignored. Use SymbolLabels with the correct `Key`.
Using `@Label.X` in SymbolLabels	`@Label.` is for symbol internals only. Use `@Tag.<path>`.
Using HMI library on operator control screens for live state	HMI is for training material and technical diagrams. Use HPG for operator screens — flat, theme-aware, calibrated for the state convention.
Trying to vary Wizard style variants (orientation, foot detail) via raw geometry	Visual variants are configured in Designer via the Wizard configuration button on the placed symbol. The AI's job is to place + wire SymbolLabels and stop.
Hardcoding symbol names that have not been verified at runtime	Always treat `list_elements()` as the authoritative catalogue. If a SymbolName does not appear there, do not write it.
Shrinking a symbol below its minimum size to fit a layout	Redesign the layout. Below minimum size the symbol's internal labels and state indicators degrade past readable.

Section 7 — Related skills

- **Skill Display Construction - Basics** — required prerequisite. Covers theme-first thinking, the build loop, binding syntax, and the global minimum-size table.
- **Skill Display Construction - Canvas** — the paradigm context for this skill. §4.5 (Choosing a symbol source) is the seed this page expands; §5 (Equipment cookbook) is where Recipe 1 (vessel) and Recipe 2 (Wizard) live.
- High Performance HMI Compliance — the philosophy behind the HPG library and the state-convention conventions used here.
- ISA-101 HMI Compliance — ISA-101 instrument-bubble conventions and the HMI symbol library's alignment to them.

Child pages, if any, are listed automatically by Confluence below.