Skill Display Construction

Router page. Display construction is split into a foundation (Basics), two paradigm specializations (Canvas, Dashboard), and a Canvas companion (HMI Symbol Choice). This page tells the AI which to load for a given task.

Parent: Platform Skills Library

When to use this skill

Use when:

- Building a new display from scratch
- User asks for a dashboard, HMI screen, or operator interface
- Need to place symbols with tag bindings
- Need visual dynamics (color changes, visibility toggles, rotation)
- Need navigation between displays
- Need CodeBehind for display logic

Do NOT use when:

- Just modifying text or properties on an existing display (use `get_objects` + `write_objects`)
- Need only a basic starter dashboard (the **New Solution** skill covers that)

Router — which child skill(s) to load

Display construction is split into focused skills. Load them based on what you're building:

Task	Load these skills
Any display work — first time in a session	**Basics** (always)
Process diagram, P&ID, HMI screen, equipment layout, spatial arrangement	Basics + **Canvas**
KPI wall, operator console, data monitoring grid, responsive card layout	Basics + **Dashboard**
Hybrid solution — some pages are P&ID, some are KPI grids	Basics + **Canvas** + **Dashboard**
Single display combining grid cells with spatial equipment imagery	Basics + **Dashboard** + **Canvas** (embed Canvas via ChildDisplay cell)
Symbol selection for vessels / pumps / valves / motors / blowers / heat exchangers / conveyors / instruments — the "which symbol do I use?" deep dive	Basics + **Canvas** + **HMI Symbol Choice**

Decision guide — Canvas vs Dashboard

If the display is primarily **data monitoring** (gauges, trends, grids, alarm viewers) → **Dashboard**.

If the display shows equipment with **spatial relationships** (pipes, process flow, custom shapes) → **Canvas**.

When in doubt, start with Dashboard — it's simpler and covers 70%+ of operator-facing displays.

When the operator question is "what does this equipment look like and what's its state?", the answer is almost always Canvas + a Wizard symbol — `Wizard/BLOWER`, `Wizard/MOTOR`, `Wizard/PUMP`, `Wizard/TANK`, or `Wizard/VALVE` — wired through SymbolLabels rather than direct properties.

The home page — wire it into the Startup layout's Content region

Every solution has a **Startup** layout (the application frame) with five fixed regions: Header, Footer, Menu, Submenu, and **Content**. The page in the **Content** region is the screen the application opens to at runtime *and* the thumbnail Solution Center renders as the solution's preview — one assignment drives both. Templates wire Content to `Display.MainPage` (the "Project home page").

So when you build the operator's first / home screen, make sure the Content region ends up holding a real, populated page — either:

- author your home content **into** the page already in Content (the template's `MainPage`), or
- point Content at your own page: read the Startup layout in full (`get_objects('DisplaysLayouts', names=['Startup'], detail='full')`), set the `Content` member's `Page` to `Display.<YourPage>` in its `Members` array, and write the whole layout back (`DisplaysLayouts` is a document object — full replacement).

Hard rule: never finish a solution with the Startup Content region pointing at an empty or placeholder page — the application opens blank *and* the Solution Center preview is blank. The preview is a cached screenshot, refreshed when that page is rendered and saved; building/publishing the solution regenerates it. Make the home page presentable — a project-name cover with at-a-glance status is a good default, since it is the operator's first impression and the preview customers see.

The child skills

Skill Display Construction - Basics

The foundation for all display work. Load this in every session that touches a display.

Covers: theme-first mental model (13 theme pairs, semantic brushes), write mechanics (`DisplaysList`, document objects, read-before-write), the build loop (`write_objects get_state fix` repeat), binding syntax and gotchas (UDT inheritance, system columns, `@Tag` vs `@Label`), spacing and typography tokens, minimum element sizes, element-category overview, symbol writing rules (Wizard, Library, Solution prefixes), common pitfalls.

Skill Display Construction - Canvas

Absolute-positioned displays for HMI, P&ID, process overviews, equipment layouts.

Covers: zone-based mental model, zone math, z-order rules, the 7 shape primitives (Rectangle, Ellipse, Polygon, Polyline, Path, Gridline, Spline), 8 first-class auto-shapes (Cylinder, Gear, Arrow, Cloud, Star, Hexagon, Pentagon, Trapezoid), 3 containers (ShapeGroup, SvgGroup, Group), the equipment cookbook (vessel with jacket, centrifugal pump, pipe segment, reactor with heating coils), the 14 dynamics types with 4 copy-paste patterns (status_indicator, toggle_button, animated_motor, level_bar), header-owns-navigation pattern, writer-side normalizations.

Skill Display Construction - Dashboard

Grid-based displays for data monitoring, KPI walls, operator consoles.

Covers: cell/grid mental model, 6 standard grid recipes (3×2 KPI wall, 4×3 operator wall, 2-column list-detail, etc.), controls-by-cell-purpose decision matrix, the TrendChart recipe (Pens forms, Duration defaults, LegendPlacement), the DataGrid listdetail pattern via UserType-typed Client tag, ComboBox zero-script FK-lookup, AlarmViewer default-template pattern, AssetsTree + ChildDisplay master-detail navigation, Carousel vs TabControl, KPI card recipes (CenterValue, composite TextBlock, threshold-colored value), Dashboard-specific quirks (cells size by content, animation dynamics ignored).

Skill Display Construction - HMI Symbol Choice

Canvas companion. Load alongside Canvas when the task is "which symbol do I use for this piece of equipment?"

Covers: the symbol source order (Solution Library/HPG Library/HMI Wizard primitives), per-equipment-class recommended path for 10 equipment families (tanks, reactors, pumps, motors, blowers, valves, heat exchangers, conveyors, instruments, process compute), sizing tables (Min / Compact / Standard / Hero) per family, SymbolLabels keys per Wizard (BLOWER, MOTOR, PUMP, TANK, VALVE) with confirmed Key / typical binding, discovery via `list_elements()`, pitfalls.

Discovery tools (common to all four)

Before building any display, use these tools to discover what's available:

```
list_elements()                    -- all element types by category + library folders
list_elements('ThemeColors')       -- all named theme brushes
list_elements('Wizard')            -- the 5 Wizard symbols
list_elements('Library/HMI')       -- browse HMI symbol library (~1600 symbols)
list_elements('Canvas')            -- Canvas display structure schema
list_elements('Dashboard')         -- Dashboard display structure schema
list_dynamics()                    -- all dynamic types by category
list_dynamics('FillColorDynamic') -- full schema for a specific dynamic
```

MCP tools and tables

Category	Items
Tools	`get_table_schema`, `write_objects`, `get_objects`, `list_elements`, `list_dynamics`, `browse_object_model`, `get_state`, `get_screenshot`, `get_display_tree`
Tables	`DisplaysList`, `DisplaysSymbols`, `DisplaysLayouts`

Critical: use `DisplaysList` for creating and editing displays. `DisplaysDraw` is the Designer visual editor UI — NOT a writable table.

Child pages are listed automatically by Confluence below.

In this section...