Amtiri

REFERENCE MANUAL

Buttons

Generated on November 25, 2026

A dashboard has two buttons. The **Button** writes its variable while it is pressed, for commands that last as long as the operator holds them. The **Action button** runs a script when it is clicked: it opens and closes panels, moves to another dashboard and shows what a script computes, but it writes no variable. This page describes both, and ends with a recipe for a Start/Stop pair that holds its command.

Tile	Widget	What a click does	Writes
Button	FieldButton	Holds its variable `true` while the mouse button is down on it	`true`, then `false`; 1, then 0 into a whole-number variable
Action Button	ActionButton	Runs its script once	Nothing

Both are HTML buttons laid over the drawing, so they are drawn over every shape whatever their place in the component list: see Paint order and layers. Both are 80 × 32 pixels when dropped, at least 30 pixels tall because of the theme, and work only in the viewer: in the editor a click selects them.

FieldButton

The **Button** tile: a momentary push button bound to a variable.

- **Pressed:** when the mouse button goes down on it, it writes `true`.
- **Released:** when the mouse button comes up on it, it writes `false`.
- **Left:** whenever the pointer leaves it, pressed or not, it writes `false`. Dragging off a pressed button releases the command.
- **Unbound,** with no Variable, it does nothing.

Each write takes the engine's own path, as an input's does: see How inputs write. Into a variable declared `int`, `long`, `short` or `byte`, such as a coil kept as a number, the button writes 1 and 0 instead of `true` and `false`. `true` and `false` fit a `boolean` variable, a `string` variable and a variable with no Type; a variable declared `double`, `float` or `decimal` refuses them. Writing `false`, or 0, to a variable that already holds it changes nothing, so the button does not disturb the plant when the pointer merely passes over it.

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`variable`	Variable	Value: a variable's code, from the picker	Text	Empty: bound to nothing	The variable held `true` while the button is pressed
`label`	Label	Formula	Text	`"Button"`	The text on the button. Evaluated in the editor and the viewer, and again whenever a variable it reads changes; inside a panel, the parameters it names are bound

- **It does not latch.** The command lasts exactly as long as the press. For a command that stays on after the operator lets go, use a FieldSwitch, which writes the opposite of what it shows at each click, or two Buttons and a latch formula, as in the recipe below.
- **A label shows text or a number.** A number is shown as its text, as on a Label: 42.5 as `42.5`. Format it for a steady look: `"%.0f rpm".format(to_double(SIT_101))`. A label formula that fails leaves the button blank.

Example. A jog button, which runs a conveyor only while it is held, with a label that says what it does:

```
FieldButton  Variable  CV201_JOG
             Label     evl(if(CV201_JOG, "Jogging", "Jog"), "Jog")
```

Styling

- **Element:** an HTML `<button>`, with the label as its text.
- **Default styles:** `{background_color:"#ccc",font_size:"24px"}`: a grey button with 24-pixel text. Because it sets `background_color`, the theme's hover and pressed tints do not show. Clear the **Styles** formula to get the theme's own button, which follows the viewer's light or dark theme.
- **Keys that work:** `background_color`, `color`, `border`, `border_color`, `border_width`, `border_radius`, `padding`, `font_family`, `font_size`, `font_weight`, `box_shadow`, `opacity`, `min_height`, `cursor`.
- **Keys to avoid:** `fill` and `stroke`, which do nothing on a button. `position`, `left`, `top`, `width` and `height`, which replace the place and size it was drawn at. `display: "none"`, which also hides it in the editor.
- **No pressed look.** Styles apply in every state; the pressed look cannot be styled. Colour the button by the plant's state instead, as below.
- **Example:** a stop button, dark red while the pump runs:

```
Styles  {background_color: evl(if(P101_RUN, "#B91C1C", "#DC2626"), "#DC2626"), color: "#FFFFFF",
         border: "none", border_radius: "6px", font_size: "16px", font_weight: 600, cursor: "pointer"}
```

Styling widgets has the rules, and a rounded button recipe.

ActionButton

The **Action Button** tile: a button that runs a script when it is clicked. The script is one Amtiri Script expression; see the language overview.

> **Warning**
>
> A script that writes a variable writes to the plant. The button's design, its roles and its testing are yours: see Safety and responsibility.

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`label`	Label	Formula	Text	`"Button"`	The text on the button. Evaluated in the editor and the viewer, and again whenever a variable it reads changes
`display`	Display result	Value: a check box	Boolean	`false`	`true` shows the script's result in a dialog
`script`	Script	Formula, run on click	Any	Empty	What the button does, run once at each click in the viewer

Inside a panel, the parameters named in the label and in the script are bound like those of every formula: see How binding works.

What a click does

1. The wait indicator covers the screen until the script has finished.
2. `_VIEW_` is set to the view the button is in, which is the panel when the button sits on a panel, and `_CALLER_` to the button.
3. The script runs once, in the engine.
4. When **Display result** is ticked and the result is not `null`, the result dialog opens.
5. `_VIEW_` and `_CALLER_` are removed again, also when the script failed.

- **Keep scripts short.** While a script runs, every dashboard view of the instance waits to redraw.
- **`_VIEW_` and `_CALLER_`** exist only while the script runs. The panel functions read them to know where to act and where to place a panel; a script has no use for them of its own, and the palette does not offer them. Give no variable either name: every click overwrites it and then removes it.

What a script can do

- **Open and close panels, and move to another dashboard,** with three functions that act only in an Action button's script: openPanel, closePanel and openDashboard. Anywhere else they do nothing. See Opening panels from a script.
- **Make several calls,** as a sequence in parentheses: `(a, b)` runs `a`, then `b`, and gives `b`'s result. See Syntax.
- **Read variables and compute,** to show the result with **Display result**.
- **It cannot write a variable.** Amtiri Script has no assignment: a formula returns a value, it does not set one. A script such as `P101_CMD = true` does not compile, and its error ends with "Amtiri Script has no assignment: a formula returns a value, it does not set variables". A dashboard writes through its inputs and its Buttons: to start a pump, bind a FieldSwitch or a FieldButton to its command.

Example. Move the screen to another dashboard:

```
Label   "Area 2"
Script  openDashboard("AREA_2_OVERVIEW")
```

Example. Close the detail panel, and open the alarm panel in its place:

```
Script  (closePanel("pump_detail"), openPanel("alarms", "AREA_2_ALARMS", 0, 180))
```

Display result

With **Display result** ticked, a result other than `null` opens a dialog titled **Result**. It shows the result as indented JSON, up to its first 10,000 characters, with three buttons: **Copy** puts the whole result on the clipboard, **Download** saves it as `data.json`, and **Cancel** closes the dialog. A script whose last call opens or closes a panel gives `null`, and no dialog opens.

Example. The readings of a pump, on demand:

```
Label           "P-101 data"
Display result  ticked
Script          {speed: SIT_101, current: IT_101, run_hours: P101_RUN_HOURS}
```

Errors

- **A script that fails,** because it does not compile or because it fails while it runs, opens a dialog titled **Unhandled Exception** with the error. The formula editor shows compile errors while the script is written.
- **Calls made before the failure keep their effect.** In `(openPanel("alarms", "AREA_2_ALARMS", 0, 180), to_int (P101_MODE))`, with `P101_MODE` holding `"AUTO"`, the panel opens and then the error shows.
- **A variable that was never written** makes a script that reads it fail: guard it with `evl`. See Errors and missing values.

Styling

- **Element:** an HTML `<button>`, with the label as its text.
- **Default styles:** `{background_color:"#ccc",font_size:"24px"}`: a grey button with 24-pixel text, without the theme's hover and pressed tints. Clear the **Styles** formula to get the theme's own button.
- **Keys:** as the FieldButton's.
- **Example:** a navigation button, in the colours of the plant's menu:

```
Styles  {background_color: "#1E3A8A", color: "#FFFFFF", border: "none", border_radius: "4px",
         font_size: "14px", font_weight: 600, padding: "0px 12px", cursor: "pointer"}
```

Recipe: a Start/Stop pair

Two Buttons, Start and Stop, and a run command that stays on after Start is released, until Stop is pressed. The Buttons write two push-button variables; a formula in the Program latches the command.

Program. Three variables:

```
Variable       Type     Initial value  Formula
P101_START_PB  boolean  false
P101_STOP_PB   boolean  false
P101_RUN_CMD   boolean  false          if(P101_STOP_PB, false, if(P101_START_PB, true, P101_RUN_CMD))
```

Dashboard. The two Buttons, and a lamp that shows the command:

```
FieldButton  Variable  P101_START_PB   Label  "Start"
FieldButton  Variable  P101_STOP_PB    Label  "Stop"
Circle       Styles    {fill: if(P101_RUN_CMD, "#16A34A", "#9CA3AF"), stroke: "#374151", stroke_width: 2}
```

How it runs. The engine calculates `P101_RUN_CMD` whenever `P101_START_PB` or `P101_STOP_PB` changes. Inside its own formula, `P101_RUN_CMD` reads the variable's previous value, and does not make the formula a trigger of itself: see Variable formulas .

The operator	`P101_START_PB`	`P101_STOP_PB`	`P101_RUN_CMD`
Has done nothing since the start	`false`	`false`	`false`
Presses Start	`true`	`false`	`true`
Releases Start	`false`	`false`	`true`: its previous value
Presses Stop	`false`	`true`	`false`
Releases Stop	`false`	`false`	`false`: its previous value
Presses Stop while Start is held, on another screen	`true`	`true`	`false`: Stop wins

- **Give all three an initial value.** A formula that reads a variable which was never written fails, with a warning, and writes nothing; the initial values make the three exist from the start.
- **After a restart** the command is `false` again, from its initial value, so the pump waits for Start. To keep the command across restarts instead, tick **Persistent** on `P101_RUN_CMD`.
- **The command reaches the plant** through whatever writes `P101_RUN_CMD` out, such as a module that forwards it to the pump's controller. Show the pump's running feedback from the plant beside the lamp, not only the command.
- **Stop overrides Start** because the formula asks for Stop first. Swap the two tests for a Start that wins.
- **A switch instead:** a single FieldSwitch bound to `P101_RUN_CMD`, with no formula on it, latches by itself. Do not bind a switch to a variable that has a formula: the formula's next result replaces what the switch wrote.

Next steps

- Inputs
- Regions and panels
- Amtiri Script overview
- Styling widgets

This page describes Data Orchester Dashboards 1.9.4.