Amtiri

REFERENCE MANUAL

Inputs

Generated on November 25, 2026

Inputs are how a dashboard writes to the plant. Each of the seven input widgets is bound to one variable: it shows that variable's value and, when the operator uses it, writes a new one through the engine, exactly as a module or a formula would. This page describes what each input writes and when, which variable types take it, and how to style it. The two buttons are on their own page, Buttons.

Tile	Widget	Draws	Writes	When
String Field	FieldText	A text box, or a text area	Text	On Enter, or when the box loses the focus
Number Field	FieldNumber	A number box	A decimal; `null` when emptied	On Enter, when the box loses the focus, and at each click of its arrows
Date Field	FieldDate	A date box with the browser's date picker	A date without a time of day; `null` when cleared	When a date is chosen
Combobox	FieldCombobox	A drop-down list	The chosen option's value, as text	When an option is chosen
Slider Horizontal	FieldSliderHorizontal	A range slider, minimum at the left	A decimal	Continuously while the thumb is dragged
Slider Vertical	FieldSliderVertical	The same slider turned upright, minimum at the bottom	A decimal	Continuously while the thumb is dragged
Switch	FieldSwitch	An on/off switch	`true` or `false`; 1 or 0 into a whole-number variable	At each click

How inputs write

- **One variable, named by its code.** An input's **Variable** holds the code of the variable it shows and writes, chosen with the picker: the instance's variables, and in a panel the panel's own parameters, marked "Panel parameter". It is a code, not a formula: an input cannot write an expression, one element of a list or one field of a record.
- **Every write takes the engine's own path,** the one a formula's result and a module's reading take:
  1. the value is converted to the **Type** the variable declares in the Program, when it declares one;
  2. it is compared with the current value, and an unchanged value stops there;
  3. it is stored, saved when the variable is **Persistent**, and logged in the server log as `CODE <- value`;
  4. the formulas and modules that read the variable, and every dashboard that shows it, hear of the change.

  See How a result becomes the value.

- **A refused write is silent for the operator.** A value the variable's Type cannot take is not written. The server log gets a warning that starts with `[TYPE]` and the variable's code and gives the reason, and the operator sees nothing: the input goes on showing what was entered until the view next redraws, and then shows the variable's value again. Check each input against its variable's Type with the table below. The instance setting `orchester.typing.strict=false` writes such values anyway, with the same warning: see the Configuration reference.

- **A variable that has never been written shows as empty:** an empty box, a switch that is off. The first value an input writes creates it. See Never-written variables.

- **Inputs work in the viewer only.** In the editor they are drawn where they will be, but cannot be operated.

Which variable types take what an input writes

The Type is the one a variable declares in the Program. A variable that declares no Type, or the Type `any`, takes every value as it comes, and every Type takes `null`.

Widget	Writes	Types that take it	Refused
FieldText	The text as typed; an emptied box writes an empty text, `""`	`string`. Number types when the text is a number with a point, such as `12.5`: `int`, `long`, `short` and `byte` only when it is whole. `boolean` for `true` or `false`. `Date` for `2026-09-29`, `DateTime` for `2026-09-29T08:00`	Text that does not convert, such as `12,5` into `double`. An empty text into any Type but `string`
FieldCombobox	The option's value, as text; the empty entry writes `""`	As FieldText	As FieldText
FieldNumber	A decimal; `null` when the box is emptied	`double`, `float`, `decimal` within its decimal places. `int`, `long`, `short` and `byte` when the number is whole and in range. `string`, as the decimal's text: `12.5`, or `12.0` for 12. `boolean` for 0 and 1 only	A fraction into a whole-number Type. Anything into `Date` or `DateTime`
FieldSliderHorizontal, FieldSliderVertical	A decimal: the minimum plus a whole number of steps	As FieldNumber. With whole-number minimum and step, every position fits an `int`	As FieldNumber: a step of 0.5 into `int` is refused at every half
FieldDate	A date without a time of day; `null` when cleared	`Date`. `DateTime`, as the start of that day. `string`, as `2026-09-29`	Number and `boolean` Types
FieldSwitch	`true` or `false`; 1 or 0 into a variable declared `int`, `long`, `short` or `byte`	`boolean`. `int`, `long`, `short` and `byte`, as 1 and 0. `string`, as `true` or `false`	`double`, `float` and `decimal`, which get `true` or `false`. Declare an on/off command `boolean` or a whole-number Type, or give it no Type

The Button writes the same values as the switch, with the same rules.

Writing one variable from several places

- **Several inputs may write one variable,** on one dashboard or on several, for one viewer or many. Each write replaces the value, the last one wins, and every view that shows the variable redraws with it.

- **An input does not own its variable.** A variable that also has a formula takes the formula's result again at its next calculation, and one that a module writes, such as a register a Modbus master reads, takes the module's next value. Bind inputs to variables that nothing else writes: a setpoint, a mode, a command.
- **To take one value from a dashboard and from another source,** such as a setpoint set on a panel and by a remote system, give each source its own variable and combine them with a Merger. It writes to one output whichever input changed last, when its filter accepts the value.

Size

- **Default size:** 80 × 32 pixels; a date field 160 × 32, wide enough to show a whole date and its picker button; a slider 160 × 32 or 32 × 160; a switch 56 × 32.
- **At least 30 pixels tall.** The theme gives one-line text boxes, number and date boxes and drop-down lists a minimum height of 30 px. Drawn smaller, the control still renders 30 px high. `min_height: "0px"` in its styles lifts the minimum. Text areas, sliders and switches take exactly the height they are drawn at. See Sizes of inputs and buttons.

Styling inputs

Inputs are HTML controls laid over the drawing, not SVG shapes, so their styles are CSS properties:

- **Keys that work:** `background_color`, `color` for the text, `border`, `border_color`, `border_width`, `border_radius`, `padding`, `font_family`, `font_size`, `font_weight`, `text_align`, `box_shadow`, `opacity`, `min_height`, `cursor`. Sizes are written with their unit, `"16px"`, and colours as text, `"#2563EB"`.
- **Keys to avoid:** `fill` and `stroke`, which do nothing on a control. `position`, `left`, `top`, `width` and `height`, which replace the place and size the widget was drawn at. `display: "none"`, which hides the input in the editor too: hide it with `visible` instead.
- **The theme.** With no styles, an input takes the look of the viewer's light or dark theme, while the canvas and its shapes keep their colours in both. Where the look matters, set `background_color`, `color` and `border_color`: the input then looks the same in both themes. A widget's styles win over the theme. An input keeps the theme's hover and focus border colours unless `border_color` is set.

Styling widgets has the rules for the styles record, colours, units and fonts.

FieldText

The **String Field** tile: a one-line text box or, with **Multiline** ticked, a text area for several lines. It shows the variable's value as text, whatever the value is: 42.5 shows as `42.5`.

It writes the text when the operator commits it: on Enter, or when the box loses the focus. A text area, where Enter starts a new line, writes when it loses the focus. An emptied box writes an empty text, `""`.

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 the box shows and writes
`multiline`	Multiline	Value: a check box	Boolean	`false`	`true` draws a text area for several lines instead of a one-line box

- **For numbers, prefer FieldNumber.** A text box can write into a number variable, but a decimal comma, a unit or a stray letter makes the engine refuse the write, silently for the operator.

Example. A shift note, kept across restarts:

```
Program     SHIFT_NOTE    Type string, Persistent
FieldText   Variable      SHIFT_NOTE
            Multiline     ticked
            Styles        {font_size: "14px", border_color: "#94A3B8", padding: "6px"}
```

Styling

- **Element:** an HTML text box, `<input type="text">`, or with Multiline a `<textarea>` that cannot be resized.
- **Default styles:** `{}` : the theme's look.
- **Keys:** those under Styling inputs; `text_align` places the text.
- **Example:** a centred, bordered box that looks the same in both themes:

```
Styles  {background_color: "#FFFFFF", color: "#0F172A", border_color: "#94A3B8",
         font_size: "16px", text_align: "center"}
```

FieldNumber

The **Number Field** tile: a number box. It shows any number the variable holds, whole or decimal.

It writes a decimal when the operator commits the number: on Enter, when the box loses the focus, and at each click of the box's arrows. Emptied, the box writes `null`, and so does anything the browser cannot read as a number. The browser hands the number over with a point for decimals, whatever the viewer's language.

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 the box shows and writes

- **Into a whole-number variable,** a whole number such as `5` is written as 5, and `5.5` is refused.
- **An emptied box writes `null`,** which every Type takes, so the formulas and modules that read the variable must allow for `null`.
- **No limits of its own.** The box takes any number. To keep a setpoint within a range, let a Merger with a filter take it from the box's variable into the real setpoint, or use a slider, which cannot leave its range.

Example. A level setpoint, taken into the real setpoint only between 0 and 4.5 m:

```
FieldNumber  Variable  SP_LIT_301_HMI
Merger       Inputs    SP_LIT_301_HMI
             Filter    _VALUE_ != null && to_double(_VALUE_) >= 0 && to_double(_VALUE_) <= 4.5
             Output    SP_LIT_301
```

Styling

- **Element:** an HTML number box, `<input type="number">`, with its text aligned right.
- **Default styles:** `{}` : the theme's look.
- **Keys:** those under Styling inputs.
- **Example:** red text while the setpoint is above the tank's alarm level:

```
Styles  {font_size: "16px", color: evl(if(SP_LIT_301_HMI > 4.0, "#DC2626", "#0F172A"), "#0F172A")}
```

FieldDate

The **Date Field** tile: a date box, which opens the browser's date picker. It shows the date the variable holds: a date, the date of a date and time, or a text written as an ISO date, `2026-09-29`, or date and time, `2026-09-29T08:00`. Any other value leaves the box empty.

It writes a date without a time of day when the operator chooses one, and `null` when the box is cleared.

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 the box shows and writes

- **Bind it to a `Date` variable.** A `DateTime` variable takes what the field writes, as the start of that day, and the field shows the date of the stored date and time; its time of day is neither shown nor kept when the operator picks another date.
- **The date is the calendar day the operator picked,** with no time of day and no time zone.
- **How the date looks** in the box, and the picker itself, are the browser's: they follow the viewer's browser and language, not the dashboard.

Example. The first day of a report, read by a formula:

```
Program    REPORT_FROM   Type Date, Persistent
FieldDate  Variable      REPORT_FROM
```

Styling

- **Element:** an HTML date box, `<input type="date">`.
- **Default styles:** `{}`: the theme's look.
- **Keys:** those under Styling inputs. The picker that opens is drawn by the browser and takes no styles.
- **Example:** a compact box, drawn 24 pixels high:

```
Styles  {font_size: "13px", padding: "2px 6px", min_height: "0px"}
```

FieldCombobox

The **Combobox** tile: a drop-down list. Its options come from a formula, so they can change with the plant, and an empty entry always comes first.

It writes the chosen option's value as text when the operator chooses one. The empty entry writes an empty text, `""`.

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 the list shows and writes
`options`	Options	Formula	List of records	`[{label:"Option 1",value:"1"},{label:"Option 2",value:"2"}]`	One record per option: `label`, the text shown, and `value`, what is written

- **Values are written as text.** A number given as a value is written as its text: `1` as `"1"`, and a decimal computed from whole numbers, such as `2 + 1`, as `"3.0"`. A number variable converts that text back, so `"1"` reaches an `int` variable as 1.
- **The list shows the variable's value** as the chosen option when the value, as text, equals one of the values. A number counts as its text as the engine writes it: an `int` variable holding 1 selects the option whose value is `"1"`, while a `double` holding 1 is `1.0` and selects only `"1.0"`. Give the options of a number variable values written the way the variable holds them, or bind the combobox to a `string` variable.
- **The options are evaluated in the editor and the viewer,** and again whenever a variable they read changes. Inside a panel, the parameters they name are bound like those of any formula.
- **The empty entry cannot be removed.** It writes `""`, which a `string` variable stores and any other Type refuses.

Example. A pump's operating mode:

```
Program        P101_MODE  Type string, Initial value "AUTO"
FieldCombobox  Variable   P101_MODE
               Options    [{label: "Automatic", value: "AUTO"}, {label: "Manual", value: "MAN"},
                           {label: "Off", value: "OFF"}]
```

Options that follow the plant, offering Manual only while the pump is in local control:

```
Options  if(evl(P101_LOCAL == true, false),
            [{label: "Manual", value: "MAN"}, {label: "Off", value: "OFF"}],
            [{label: "Automatic", value: "AUTO"}, {label: "Off", value: "OFF"}])
```

Styling

- **Element:** an HTML drop-down list, `<select>`, with one `<option>` per entry.
- **Default styles:** `{}`: the theme's look.
- **Keys:** those under Styling inputs. The list that opens is drawn by the browser and takes no styles.
- **Example:** an amber list while the pump is not in Automatic:

```
Styles  {background_color: evl(if(P101_MODE == "AUTO", "#FFFFFF", "#FEF3C7"), "#FFFFFF"),
         border_color: "#D97706", color: "#78350F"}
```

FieldSliderHorizontal

The **Slider Horizontal** tile: a range slider with its minimum at the left. It shows any number the variable holds, whole or decimal, and the track fills with the accent colour up to the thumb.

It writes while the thumb is dragged: the latest position is sent each time the server has answered the one before, and the position where the thumb stops is always written. A click on the track writes that position.

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 the slider shows and writes
`min`	Min	Value: a number box	Number	`0`	The value at the left end
`max`	Max	Value: a number box	Number	`100`	The value at the right end
`step`	Step	Value: a number box	Number	`1`	The distance between two positions of the thumb: the slider writes the minimum plus a whole number of steps

- **Every drag writes many values.** Each one is a write, with everything a write brings: a log line, the formulas and modules that read the variable, and a redraw of every view that shows it. Bind sliders to setpoints that can take intermediate values; for a value that must change in one step, use a number box.
- **`min`, `max` and `step` are values,** not formulas: the range cannot follow the plant.
- **A value outside the range** is shown at the nearer end; the slider never writes it.
- **Size.** The thumb is as thick as the box, at least 10 pixels, and the track 8/18 of the thumb, at least 3 pixels. A slider drawn large is a large slider, easy to hit on a wall screen.

Example. A pump speed setpoint from 20 to 50 Hz, in steps of 0.5:

```
FieldSliderHorizontal  Variable  SP_SIT_101
                       Min 20    Max 50    Step 0.5
                       Styles    {__j2w_accent: "#0891B2"}
```

Styling

- **Element:** an HTML range slider, `<input type="range">`.
- **Default styles:** `{}`: the theme's accent colour for the filled part and the thumb.
- **Keys:** `__j2w_accent`, the accent colour, is the only key that reaches the filled part of the track and the thumb. `opacity` works too.
- **Keys to avoid:** the slider's own `__j2w_slider_...` properties, which it sets itself after the styles, and those listed under Styling inputs.
- **Example:** red while the speed is above its limit:

```
Styles  {__j2w_accent: evl(if(SP_SIT_101 > 45, "#DC2626", "#0891B2"), "#0891B2")}
```

FieldSliderVertical

The **Slider Vertical** tile: the same slider turned a quarter turn, with its minimum at the bottom. It shows, writes and sizes itself as FieldSliderHorizontal does; its thickness is the width of its box.

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 the slider shows and writes
`min`	Min	Value: a number box	Number	`0`	The value at the bottom end
`max`	Max	Value: a number box	Number	`100`	The value at the top end
`step`	Step	Value: a number box	Number	`1`	The distance between two positions of the thumb

Example. A valve opening from 0 to 100 %, beside a tank:

```
FieldSliderVertical  Variable  SP_FCV_201
                     Min 0     Max 100    Step 1
```

Styling

- **Element:** an HTML range slider, `<input type="range">`, turned with `transform: rotate(-90deg)` and laid out with its sides swapped so that, turned, it covers the box it was drawn in.
- **Default styles:** `{}` : the theme's accent colour.
- **Keys:** `__j2w_accent` and `opacity`, as on the horizontal slider.
- **Keys to avoid:** `transform`, which replaces the turn and lays the slider flat across its box. The slider's own `__j2w_slider_...` properties, and those listed under Styling inputs.
- **Example:** `{__j2w_accent: "#16A34A"}`.

FieldSwitch

The **Switch** tile: an on/off switch that latches. It shows what the variable is now, and a click writes the opposite: `false` when it shows on, `true` when it shows off. The knob slides between the two states.

It shows **on** for `true`, for any number but 0, and for any text but `""`, `"0"` and `"false"`. It shows **off** for `false`, `0`, those texts, and a variable that has never been written. In the editor it always shows off.

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 the switch shows and writes

- **It writes a Boolean, or 1 and 0.** Into a variable declared `int`, `long`, `short` or `byte`, such as a coil kept as a number, it writes 1 for on and 0 for off; into any other variable, `true` and `false`. A variable declared `double`, `float` or `decimal` refuses the Boolean, so a switch bound to one shows the plant's state but cannot change it. See the type table.
- **Bound to nothing, it writes nothing.**
- **For a command held while a button is pressed,** use the Button, which is momentary.

Example. A pump's local run command, green when on:

```
FieldSwitch  Variable  P101_CMD
             Styles    {background_true: "#16A34A", background_false: "#9CA3AF"}
```

Styling

- **Element:** an HTML toggle, `<togglebutton class="amtiri-switch">`, drawn as a pill with a round knob. It keeps the height it is drawn at: the 30-pixel minimum does not apply.
- **Default styles:** `{}`: the theme's accent colour when on, its sunken surface when off.
- **Keys the switch reads itself:**

Key	What the switch does with it	When it is left out
`background_true`	Colours the track, and its border, while the switch is on	The theme's accent colour
`background_false`	Colours the track while the switch is off	The theme's sunken surface

- **Other keys:** `opacity`, and `cursor`.
- **Keys to avoid:** `background_color`, which competes with the two state colours, and those listed under Styling inputs.
- **Example:** the track amber while an interlock holds the pump, green otherwise:

```
Styles  {background_true: evl(if(P101_INTERLOCK, "#D97706", "#16A34A"), "#16A34A"),
         background_false: "#9CA3AF"}
```

Next steps

- Buttons
- Regions and panels
- Styling widgets
- Logic modules

This page describes Data Orchester Dashboards 1.9.4.