Amtiri

REFERENCE MANUAL

Properties every widget shares

Generated on November 25, 2026

Every widget, whatever it draws, has a name and a place on the canvas, can be hidden, takes a style, and can be moved or resized by the plant while the dashboard is shown. This page describes those shared properties, how widgets are placed and sized in the editor, and the order they are drawn in. Each widget's own properties are on its family's page: Shapes and text, Images, Gauges, Charts, Inputs, Buttons, Regions and panels and Groups.

The shared properties

The setup panel, at the right of the editor, shows the properties of the selected widget. A property is either a value, set once, or a formula, evaluated each time the dashboard redraws.

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`name`	Name	Value	Text	The type and a number, such as `Rect 4`	The widget's name in the component list. A Region's name is also the id of the panel it opens
`x`	—	Value	Number	Where the widget was dropped	Horizontal position in canvas pixels, from the left edge: a box's left side, a Label's anchor point, the start of a Line or of a Labels widget
`y`	—	Value	Number	Where the widget was dropped	Vertical position in canvas pixels, from the top edge: a box's top side, a Label's baseline, the start of a Line or of a Labels widget
`width`	—	Value	Number	80, unless the widget's page says otherwise	Width of the widget's box, in canvas pixels
`height`	—	Value	Number	32, unless the widget's page says otherwise	Height of the widget's box, in canvas pixels
`visible`	Visible	Formula, viewer only	Boolean	Empty: shown	`false` hides the widget in the viewer. Anything that is not a Boolean shows it
`styles`	Styles	Formula	Record	The widget's own, on its page	The widget's look, as a record of style keys. See Styling widgets
`dX`	X offset	Formula, viewer only	Number	Empty: 0	Added to `x` in the viewer
`dY`	Y offset	Formula, viewer only	Number	Empty: 0	Added to `y` in the viewer
`dWidth`	Width offset	Formula, viewer only	Number	Empty: 0	Added to `width` in the viewer
`dHeight`	Height offset	Formula, viewer only	Number	Empty: 0	Added to `height` in the viewer

- **A box, or points.** `width`, `height`, `dWidth` and `dHeight` belong to the widgets drawn in a box: every widget except Label, Labels, Line and a group. A Label is placed by its anchor point; a Line and a Labels widget by their two ends, the second of which is described on their page.
- **The canvas's coordinates** start at its top-left corner, and `y` grows downwards.
- **Every stored widget also carries its `type`,** such as `Rect` or `FieldNumber`. The editor writes it, and it is never edited.
- **Names need not be unique,** except for Regions: two regions with the same name share one panel.

Placing and sizing

Position and size have no fields: a widget is placed and sized on the canvas.

- **Dropping** a widget from the palette places it at the pointer: a box is centred on it, a Label's anchor point is on it, and a Line or Labels widget is centred on it.

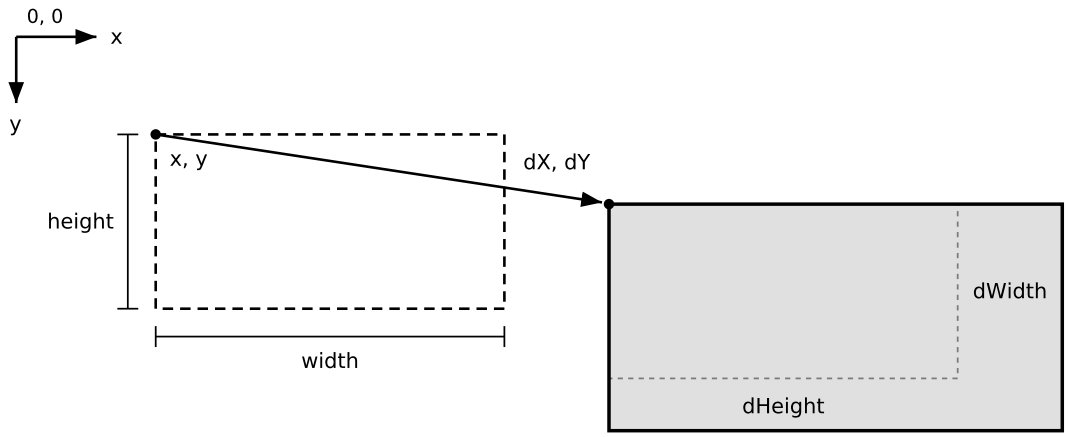
- **Handles** appear on the selected widget. A box has one at each corner, which resizes it, and one at its centre, which moves it. A Label has one, at its anchor point. A Line and a Labels widget have one at each end and one in the middle. A group has one, at its centre.
- **Holding Shift** while dragging a handle or dropping a widget snaps it to an 8-pixel grid.
- **The arrow keys** move the selection by 1 pixel, and by 8 pixels onto the grid with Shift.
- **Several widgets** selected together each get a centre handle, and dragging any of them moves them all.

The toolbar above the canvas arranges a selection of two widgets or more. Holding Shift while clicking a button snaps the result to the grid.

Buttons	What they do
Align left, **Centre horizontally**, **Align right**	Line the widgets up on the left side, the middle or the right side of the box around the selection
Align top, **Centre vertically**, **Align bottom**	The same, vertically
Expand left, **Expand right**, **Expand up**, **Expand down**	Stretch each box to that side of the box around the selection. A Line or Labels widget moves its end on that side
Distribute evenly, horizontally and **vertically**	Space the widgets' centres evenly between the two outermost
Distribute by weight, horizontally and **vertically**	Leave equal gaps between the widgets, whatever their sizes. Needs three widgets or more

Offsets

`x`, `y`, `width` and `height` are fixed: they are where the widget was placed. The four offsets are formulas that the viewer **adds** to them, so the plant can move and resize a widget:



Dashed: the box as placed in the editor, at x, y with its width and height.
Solid: the box the viewer draws, at x + dX, y + dY, with width + dWidth and height + dHeight.

- **An empty or failing offset counts as 0,** and a negative one moves the widget left or up, or makes it smaller.
- **Offsets are evaluated in the viewer only.** The editor draws the widget where it was placed, so it can always be found and selected there.
- **An offset must return a number,** whole or decimal.
- A Line and a Labels widget also offset their second end, with **To X offset** (`dToX`) and **To Y offset** (`dToY`).
- A Region's **Popup dX** and **Popup dY** are offsets too, but of the panel it opens, measured from the region's centre: see Region.

Example. A level bar that grows upwards from the bottom of a tank. The Rect is drawn over the whole inside of the tank, 200 pixels high, so the editor shows the full tank; the offsets take away the empty part at the top:

```
Rect      x 400, y 100, width 60, height 200
Y offset       200 * (1 - clamp(to_double(LIT_301) / 4.5, 0.0, 1.0))
Height offset  -200 * (1 - clamp(to_double(LIT_301) / 4.5, 0.0, 1.0))
```

With `LIT_301` at 2.7 m of a 4.5 m tank, the bar is 120 pixels high and its top is 80 pixels below the tank's top. Styling widgets gives it a colour that follows the level.

> **Important**
>
> The gauges work differently. On RadialGauge, RadialLabels, HorizontalGauge and VerticalGauge, the formulas for their values and angles, such as **Value 2** (`dValue2`) or **Start angle** (`dStartAngle`), **replace** a fixed value instead of adding to it, fall back to that fixed value when they are empty or fail, and are evaluated in the editor as well. RadialGauge and RadialLabels show no fields for the four offsets. See Gauges.

Result types

A formula must return the type its property expects. One that returns anything else is treated as a formula that fails, silently: the property falls back as described in When a formula fails.

- `visible` **needs a Boolean.** `true` shows the widget and `false` hides it; `1` , `0` and `"false"` are not Booleans, and the widget stays visible. Write a comparison, guarded for a variable that may never have been written: `evl(P101_RUN == true, false)` , `evl(COIL_12 != 0, false)` , `evl(LIT_301 > 0.2, false)` .
- **The offsets need a number,** whole or decimal. A text such as `"12"` counts as 0.
- `styles` **needs a record,** written in braces: `{fill: "#2563EB"}` .
- **A label shows text.** A number is shown as its text: 42.5 as `42.5` , 3 as `3` , and a decimal computed from whole numbers, such as `2 + 1` , as `3.0` . To fix the number of decimals, format it: `"%.1f".format(to_double(TIT_201))` . The decimal separator follows the instance's locale.
- **Lists** hold what their widget expects: texts or numbers for the labels of a Labels widget, numbers for a chart's values, records with a label and a value for a Combobox's options.

Order

The component list, in the editor's left column, lists the widgets in the order they are drawn: **the first is on top**. A group's row holds its members, in their own order.

- **Move up** and **Move down**, the arrow buttons above the list, move the selected widgets one place. Moving up draws a widget over one more of its neighbours. Inside a group, a widget moves among the group's members; a selection that spans a group and the canvas does not move.
- **A dropped widget** goes to the top of the list. **A pasted one** goes just above the widget that was selected, in the same group, or to the top of the list when nothing was selected. **A new group** takes the place of the highest of the widgets it gathers.
- Inputs and buttons are drawn over every shape whatever their place in the list, and a Region needs to be above what it covers: see Paint order and layers.

Selecting, copying and grouping

- **Click** a widget to select it, **Shift+click** to add one, and **Ctrl+click** (Cmd+click on a Mac) to add or remove one. Dragging on an empty part of the canvas draws a band that selects every top-level widget whose centre is inside it.
- **A click on a group** selects the whole group; a second click goes one level in, to the member under the pointer. Ctrl+click (Cmd+click) goes straight to the member.
- **In the list,** Shift+click selects a run of rows, and Ctrl+click (Cmd+click) adds or removes one.

Keys (Cmd instead of Ctrl on a Mac)	Action
Ctrl+A	Select every widget at the top level
Ctrl+C	Copy the selection
Ctrl+V	Paste
Ctrl+G	Group the selection
Ctrl+U	Ungroup the selected groups
Ctrl+Z	Undo
Ctrl+Shift+Z or Ctrl+Y	Redo
Delete or Backspace	Delete the selection
Arrow keys	Move the selection 1 pixel
Shift+arrow keys	Move the selection 8 pixels, onto the grid

- **Paste** puts the copies' top-left corner 16 pixels right of and below the first selected widget's, or places the copies 32 pixels from the originals when nothing is selected, and keeps them inside the canvas. They get new names. The copy is kept by the session, so widgets can also be pasted into another dashboard or panel opened in the same workspace.
- **Group** needs two widgets or more, none of them inside a group already. **Ungroup** puts the members back where the group was. Deleting a group's last member deletes the group. See Groups.
- **Undo** and **Redo** cover every edit, including the setup panel's check boxes and the picture settings of Image and SVG. Their buttons name the action they would undo or redo.

Styles

`styles` is a formula that returns a record of style keys, such as `{fill: "#2563EB", stroke: "#1E3A8A", stroke_width: 2}`. The widget writes them onto what it draws, with every `_` in a key written as `-`, so `stroke_width` becomes the property `stroke-width`. Shapes and texts take SVG properties such as `fill` and `stroke`; inputs and buttons take CSS properties such as `background_color` and `color`. The gauges style their value with a second formula, `valueStyles`, and a radial gauge's `styles` draws its track. The SVG widget, the horizontal and vertical gauges and groups take no styles, and their setup panels offer no Styles field. Styling widgets has the rules, the colours and the recipes.

Next steps

- Styling widgets
- How dashboards draw and refresh
- Shapes and text
- Gauges

This page describes Data Orchester Dashboards 1.9.4.