Amtiri

REFERENCE MANUAL

Charts

Generated on November 25, 2026

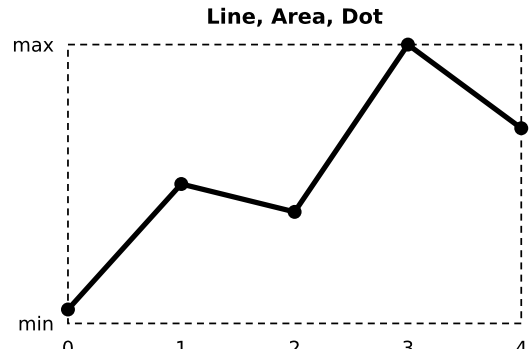
Six widgets draw a list of numbers inside their box: four charts and two grids. They draw the data and nothing else: no axes, ticks, captions or legends. A finished chart is composed from several widgets, the chart itself, a grid behind it and Distributed labels for its scale, all given the same box.

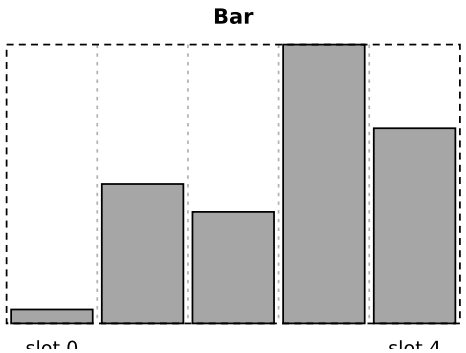
Widget	Palette tile	Stored type	Draws	Typical use
Line chart	Line Chart	`ChartLine`	A line through the numbers	A trend over the last hours
Area chart	Area Chart	`ChartArea`	The area under that line, filled	A level or a load over time
Bar chart	Bar Chart	`ChartBar`	One bar per number	Hourly or daily totals
Dot chart	Dot Chart	`ChartDot`	One dot per number	Single readings, samples
Horizontal grid	Horizontal Grid	`ChartGridH`	A horizontal line at each value of the list	The value lines behind a chart
Vertical grid	Vertical Grid	`ChartGridV`	A vertical line at each value of the list	The time lines behind a chart

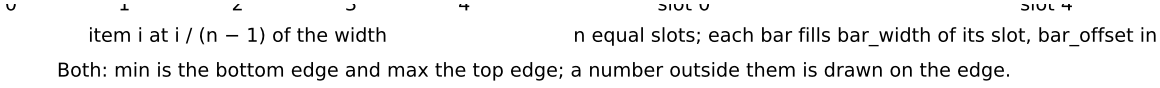
The charts also have the properties every widget shares: see Properties every widget shares. Their own three are the same on all six, and are formulas the editor evaluates too, so the editor draws the plant's live data.

Values, min and max

- **Values** is a list of numbers, such as `[5, 50, 40, 100, 70]`. The numbers are placed by their position in the list, at equal steps across the box, never by a time: a list for a chart holds one number per equal step, one per hour for example.
- **Min** is the value drawn at the bottom edge of the box and **Max** the value drawn at its top edge. A number outside them is drawn on the edge, so a chart never leaves its box. With Min above Max the chart is drawn upside down; with both equal every number sits on the middle line. Left empty, or when its formula fails, Min counts as 0 and Max as 100.
- **An item that is not a number,** such as null or a text, is skipped. A Values formula that fails draws nothing.
- **One style for all the data.** A chart takes one styles record, and every line, bar or dot it draws gets it: one colour per chart. For several series, lay several charts over each other with the same box, Min and Max.
- **No timer.** A chart is redrawn when its view repaints, and a view repaints when a variable one of its formulas names changes. A chart computed from a logger's history names the variable as text, not as a variable, so nothing would ever redraw it: keep the list in a variable instead, recalculated by a Scheduler, as in Charting a logger's history.







Line chart

A line chart draws a line through its numbers: item i of n sits i/(n # 1) of the way across the box, so the first number is on the left edge and the last on the right edge. It needs at least two numbers: with fewer it draws nothing, rather than keeping its last line. A new line chart is 80 × 32 px.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[5, 50, 40, 100, 70]`	The numbers to draw, left to right
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the bottom edge. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the top edge. Empty counts as 100

Styling

- **Element**: one SVG `<polyline>`. `styles` applies to it.
- **Default styles**: `{fill:"none",stroke:"#000",stroke_width:4}`.
- **Keys that work**: `stroke`, `stroke_width`, `stroke_opacity`, `stroke_dasharray` (as text, `"6 3"`), `stroke_linejoin`, `stroke_linecap` (round by default, as for every line on the canvas) and `opacity`.
- **Keys to avoid**: any `fill` but `"none"`: a fill paints the area between the line and the straight line joining its ends. Use an Area chart for a filled area.

The widget reads no style key itself.

Styled example. A thin blue trend with rounded corners:

```
Styles   {fill:"none",stroke:"#2563EB",stroke_width:2,stroke_linejoin:"round"}
```

Area chart

An area chart fills the shape that runs from the bottom-left corner of its box up through its numbers, placed as a line chart places them, and down to the bottom-right corner. It fills down to the bottom edge, which is Min, not down to zero. It needs at least two numbers: with fewer it draws nothing. A new area chart is 80 × 32 px.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[5, 50, 40, 100, 70]`	The numbers to draw, left to right
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the bottom edge, where the area ends. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the top edge. Empty counts as 100

Styling

- **Element**: one SVG `<polyline>`, filled. `styles` applies to it.
- **Default styles**: `{fill:"#6CF",stroke:"none",stroke_width:0}`.
- **Keys that work**: `fill`, `fill_opacity`, `stroke`, `stroke_width`, `stroke_opacity` and `opacity`. A stroke outlines the left edge, the line through the numbers and the right edge, but not the bottom edge.
- **Keys to avoid**: none in particular. For a line along the top of the area, lay a line chart with the same values over it rather than stroking the area.

The widget reads no style key itself.

Styled example. A pale area under a line chart that shows the same list:

```
Styles   {fill:"#2563EB",fill_opacity:0.15,stroke:"none"}
```

Bar chart

A bar chart cuts its box into as many equal slots as it has numbers, and draws one bar in each: the bar rises from the bottom edge, which is Min, to its number. Across its slot, the bar starts `bar_offset` of the slot's width in and is `bar_width` of it wide; with the defaults, 0.05 and 0.9, it is centred with a small gap on each side. A number at or below Min draws a bar of no height. A new bar chart is 80 × 32 px.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[5, 50, 40, 100, 70]`	The numbers to draw, one bar each, left to right
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the bottom edge, where the bars start. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the top edge. Empty counts as 100

Styling

- **Element**: an SVG `<g>` holding one `<rect>` per number. `styles` applies to every bar alike.
- **Default styles**: `{fill:"#6CF",stroke:"#ddd",stroke_width:1,bar_width:0.9,bar_offset:0.05,rx:"4px",ry:"4px"}`.

- **Keys that work**: `fill`, `fill_opacity`, `stroke`, `stroke_width`, `stroke_opacity`, `rx` and `ry` (rounded corners), `opacity`, and the two keys below. `rx` and `ry` are lengths: a number is written in pixels, and a text must carry its unit.
- **Keys to avoid**: none in particular. `bar_width` and `bar_offset` are read as numbers or as numeric text, such as `"0.6"`; anything else counts as left out.

The widget reads two style keys itself. Both are also written into the bars' style, where browsers ignore them.

Key	What the widget does with it
`bar_width`	The width of each bar, as a share of its slot. Without it, or when it is not a number, 0.9
`bar_offset`	The gap before each bar, as a share of its slot. Without it, or when it is not a number, 0.05

Styled example. Narrower bars, centred in their slots (0.2 + 0.6 + 0.2 = 1), with square corners:

```
Styles   {fill:"#0EA5E9",stroke:"none",bar_width:0.6,bar_offset:0.2,rx:0,ry:0}
```

Dot chart

A dot chart draws one dot per number, placed as a line chart places its points: item i of n at i/(n # 1) of the way across, the first on the left edge and the last on the right edge. A single number sits on the left edge. The dots' radius comes from the style key `r`. A new dot chart is 80 × 32 px.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[5, 50, 40, 100, 70]`	The numbers to draw, one dot each, left to right
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the bottom edge. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the top edge. Empty counts as 100

Styling

- **Element**: an SVG `<g>` holding one `<circle>` per number. `styles` applies to every dot alike.
- **Default styles**: `{fill:"#6CF",stroke:"#ddd",stroke_width:3,r:6}`.
- **Keys that work**: `fill`, `fill_opacity`, `stroke`, `stroke_width`, `stroke_opacity`, `opacity`, and `r`.
- **Keys to avoid**: none in particular. `r` is read as a number of pixels or as numeric text, with or without `px`: `r:3` and `r:"3px"` draw the same dot.

The widget reads one style key itself:

Key	What the widget does with it
`r`	The radius of each dot, in pixels. Without it, or when it is not a number, 6

Styled example. Small solid dots:

```
Styles   {fill:"#7C3AED",stroke:"none",r:3}
```

Horizontal grid

A horizontal grid draws a line across the whole width of its box at each value of its list, placed as the charts place their numbers: Min at the bottom edge, Max at the top edge. Give it the chart's box, Min and Max, and it marks the chart's values. A new grid is 80 × 32 px, with lines at `[0, 25, 50, 75, 100]`.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[0, 25, 50, 75, 100]`	The values to draw a horizontal line at
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the bottom edge. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the top edge. Empty counts as 100

Styling

- **Element**: an SVG `<g>` holding one `<line>` per value. `styles` applies to every line alike.
- **Default styles**: `{stroke:"#888",stroke_width:1}`.
- **Keys that work**: `stroke`, `stroke_width`, `stroke_opacity`, `stroke_dasharray` (as text) and `opacity`.
- **Keys to avoid**: `fill`, which a line does not use.

The widget reads no style key itself.

Styled example. Light dashed lines:

```
Styles   {stroke:"#D1D5DB",stroke_width:1,stroke_dasharray:"4 4"}
```

Vertical grid

A vertical grid draws a line across the whole height of its box at each value of its list. Here Min is the value at the left edge and Max the value at the right edge. A new grid is 80 × 32 px, with lines at `[0, 25, 50, 75, 100]`.

- **To mark the items of a line, area or dot chart** of n numbers, set Min to 0 and Max to n # 1: the value i then falls on item i. For 24 hourly numbers, Min 0, Max 23 and the values `[0, 6, 12, 18, 23]` draw five lines.
- **To mark the slots of a bar chart** of n numbers, set Min to 0 and Max to n: the value i then falls on the left edge of slot i.

Properties

Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`values`	Values	Formula; the editor evaluates it too	List of numbers	`[0, 25, 50, 75, 100]`	The values to draw a vertical line at
`min`	Min	Formula; the editor evaluates it too	Number	`0.0`	The value at the left edge. Empty counts as 0
`max`	Max	Formula; the editor evaluates it too	Number	`100.0`	The value at the right edge. Empty counts as 100

Styling

- **Element**: an SVG `<g>` holding one `<line>` per value. `styles` applies to every line alike.
- **Default styles**: `{stroke:"#888",stroke_width:1}`.
- **Keys that work**: `stroke`, `stroke_width`, `stroke_opacity`, `stroke_dasharray` (as text) and `opacity`.
- **Keys to avoid**: `fill`, which a line does not use.

The widget reads no style key itself.

Styled example. A marker for the current shift change, thicker than the other lines:

```
Styles   {stroke:"#F59E0B",stroke_width:2}
```

Charting a logger's history

A chart of the last 24 hours of a temperature, one point per hour, takes three pieces: a variable that holds the 24 hourly averages, a Scheduler that brings it up to date every hour, and a chart that draws the variable.

The variable. Leave its type empty, so that it takes the list as the formula gives it, and write this formula:

```
Variable   TIT_101_24H
Formula    with(now(), window_end,
             forEach(
               tsSplit(
                 history(PLANT_LOG, "TIT_101", addHours(window_end, -24), window_end),
                 psplit(addHours(window_end, -24), window_end, 24)),
               bucket,
               tsAverage(bucket)))
```

Read from the inside out:

1. `with(now(), window_end, …)` fixes the end of the window once, so every part of the formula uses the same instant. `now()` is in UTC, like the times the logger stores.
2. `history` `(PLANT_LOG, "TIT_101", from, to)` gives every sample the LOGGER `PLANT_LOG` holds for `TIT_101` over the window, oldest first. It starts with the value in force at `from`, when the logger held one then, and ends with the last value repeated at `to`. The logger's code is written bare, the variable's code in quotes.
3. `psplit` `(from, to, 24)` cuts the window into 24 equal periods of one hour, as a list of start and end pairs.

4. `tsSplit(samples, periods)` cuts the samples into one series per period. Each series starts with the value in force at the start of its hour and ends with the value in force at its end.
5. `forEach(list, bucket, tsAverage(bucket))` replaces each hour's series by its `tsAverage`, the average weighted by time: a value held for 40 minutes of the hour counts for 40 minutes. The result is a list of 24 numbers, the oldest hour first.

`window_end` and `bucket` are names the formula binds while it runs; choose names that no variable uses.

The Scheduler. One definition of a Scheduler recalculates the variable at the start of every hour:

```
Variable   TIT_101_24H
CRON       0 0 * * * ?
```

The chart. A line chart that draws the variable, over a horizontal grid with the same box, Min and Max:

```
Line chart        Values TIT_101_24H    Min 0    Max 120    Styles {fill:"none",stroke:"#DC2626",stroke_width:2
Horizontal grid   Values [0, 30, 60, 90, 120]    Min 0    Max 120    Styles {stroke:"#E5E7EB",stroke_width:1}
```

Place the grid after the chart in the widget list, so that the line is drawn over it. A Distributed labels widget from the bottom-left corner to the top-left corner, with `["0", "30", "60", "90", "120 °C"]` and `text_anchor:"end"`, writes the scale beside it, and a second one under the chart, with `["24 h ago", "last hour"]`, marks its two ends.

- **Only the Scheduler recalculates the list.** The formula names `TIT_101` in quotes, so the temperature's changes do not recalculate it. The names it uses bare, `PLANT_LOG`, `window_end` and `bucket`, are not variables either: when the engine starts, its log warns that these triggers of `TIT_101_24H` are not defined, and nothing else comes of it.
- **Until the first tick** the variable holds no value, and the chart draws nothing. Make the variable persistent to keep the last list across a restart.
- **Hours before the logger held anything** average to no value, and the chart skips them. A variable the logger never recorded gives an empty list.
- **Do not put `history` in the chart's Values.** It would read the logger at every repaint of the view, each time any variable on the view changes, and still never redraw the chart on its own.

Next steps

- Logger
- Scheduler
- `history` and `tsAverage`
- Shapes and text
- Styling widgets

This page describes Data Orchester Dashboards 1.9.4.