Amtiri

REFERENCE MANUAL

Regions and panels

Generated on November 25, 2026

A panel is a small dashboard that opens over another one: the faceplate of a pump, the detail of a tank, a list of alarms. It is written once, with **parameters** in place of variables, and bound to real variables each time it opens, so one faceplate serves every pump of a plant. A **Region** opens a panel when an area of the drawing is clicked or pointed at; an Action button's script can open one too. This page describes panels, the Region, how binding works, and how panels close.

Piece	What it is	Described in
Panel	A dashboard of category Panel, which declares parameters	Panel dashboards
Region	An invisible area that opens a panel on a click or when the pointer moves over it	Region
`openPanel`, `closePanel`, `openDashboard`	Functions for an Action button's script	Opening panels from a script

Panel dashboards

A panel is a dashboard whose **Category** is **Panel**. It is built in the same editor as every dashboard, with the same widgets, and its settings mean this:

Setting	For a panel
Code	What a Region and the script functions name to open it
Category	**Panel**. The other choice, **Dashboard**, makes it a screen of its own
Parameters	The names its widgets use in place of variables. The list is shown for a panel only
Width, **Height**	The size of the panel on screen when it opens, in canvas pixels
Background	The colour behind its drawing. Below 100 % opacity, the dashboard underneath shows through

- **Parameters** are names typed into the **Parameters** list, such as `PUMP_RUN` and `PUMP_SPEED`. Each one is then offered in every formula and every variable picker of the panel, marked "Panel parameter", and a widget uses it as if it were a variable. Whoever opens the panel says which variable each parameter stands for.
- **In the editor the parameters are not bound.** A formula that reads one reads a variable of that name, so the panel's widgets show what they show when a formula fails, unless such a variable exists.
- **Saving the panel as a Dashboard** clears its parameters.
- **A panel is only ever opened by another view.** Panels are listed with the dashboards in the configuration, marked by their icon, but never in the operator's menu, on the phone, or as a user's default dashboard.
- Saving a panel needs the `CONFIGURATION` role, as for every dashboard.

Region

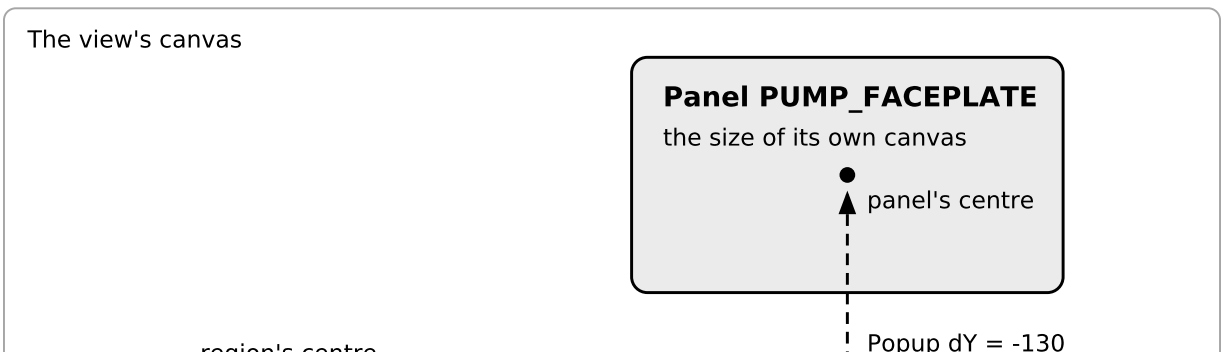
The **Region** tile: an invisible rectangle laid over the drawing, over a pump in a plant diagram for example, that opens one panel, bound to that pump's variables, when it is clicked or when the pointer moves over it. A Region carries no script: an Action button is for anything else.

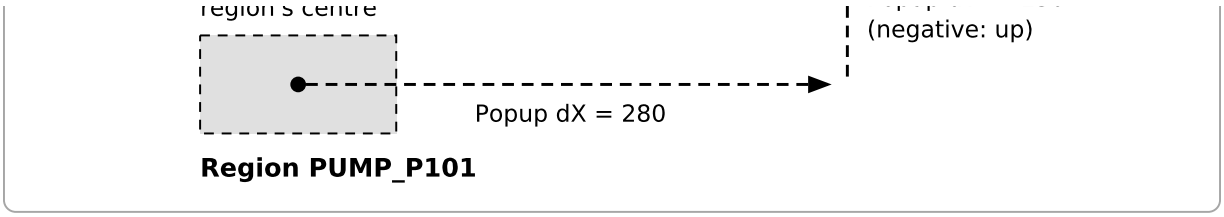
Key	Editor label (English)	Formula or value	Result type	Default	Meaning
`trigger`	Trigger	Value: **On click** or **On hover**	`CLICK` or `HOVER`	`CLICK`	What opens the panel, and therefore what closes it: see Closing
`panelCode`	Panel	Value: chosen from the panels, by name	Text: a panel's code	Empty: no panel	The panel the region opens
`offsetX`	Popup dX	Formula, viewer only	Number	Empty: 0	Horizontal distance, in canvas pixels, from the region's centre to the panel's centre
`offsetY`	Popup dY	Formula, viewer only	Number	Empty: 0	Vertical distance from the region's centre to the panel's centre. Negative places the panel above
`parameters`	Parameters, then one row per parameter of the panel	Value: a variable picker per parameter	Record: `{PARAMETER: "CODE"}`	`{}`	The variable each of the panel's parameters stands for

- **The region's name is the panel's id.** A screen shows one panel per id. Opening the same panel, with the same bindings, in the same place again changes nothing, which keeps a region **On hover** from flickering; anything else replaces what the id shows. Two regions with the same name therefore share one panel. Rename the dropped `Region 3` to something like `PUMP_P101`: the name is also what closePanel takes.
- **Choosing another panel** clears the bindings, and the rows below show the new panel's parameters.
- **A row left empty** leaves that parameter unbound: see Unbound parameters.
- **A Region is clicked only where nothing is drawn over it.** Keep it above the shapes it covers in the component list. It can never cover an input or a button, which are drawn over every shape: see Paint order and layers.
- **A panel that is not there.** When the Panel names no panel, because the panel was deleted or its code changed, nothing opens, and the server log names the region.
- A Region is 80 × 32 pixels when dropped.

Where the panel opens

The panel's centre is the region's centre moved by **Popup dX** and **Popup dY**, in the canvas pixels of the view the region is in. The region's centre is where the region is drawn in the viewer, after its own offsets. The panel takes the size of its own canvas.





- **The offsets are evaluated in the viewer only,** at each redraw, so they can follow the plant. The editor draws them as 0.
- **Nothing keeps a panel inside the canvas.** Choose offsets that keep the whole panel on screen.
- **The panel is drawn in the top layer,** over every widget, at the zoom of the view it opens on.

Styling

- **Element:** an SVG `<rect>`, almost transparent: invisible, but it takes clicks.
- **Default styles:** `{fill:"#000",fill_opacity:0.001,stroke_opacity:0.001}`. In the viewer the pointer becomes a hand over it.
- **The key the region reads itself:**

Key	What the region does with it	When it is left out
`cursor`	The pointer's shape over the region, such as `"pointer"`, `"help"` or `"default"`	`pointer`, a hand

- **Keys that work:** `cursor`; and `fill`, `fill_opacity`, `stroke` and `stroke_width` to see the region while it is placed.
- **Keys to avoid:** `fill: "none"`, `visibility: "hidden"` and `display: "none"`. A region that is not painted cannot be clicked.
- **Example:** a region shown in blue while it is placed. Put the default styles back once it sits where it should:

```
Styles  {fill: "#2563EB", fill_opacity: 0.25, stroke: "#2563EB", stroke_width: 1}
```

How binding works

When a panel opens, a copy of its drawing is rewritten, as text, before it is drawn:

1. The **Variable** of every input and Button that names a parameter is replaced by the code bound to it.
2. In every formula of every widget, each name equal to a parameter is replaced by the code bound to it: labels, styles, `visible`, offsets, values, options, an Action button's label and script. Names match without regard to case, as variable codes do.
3. Groups are rewritten member by member.

The panel then reads and writes real variables, and redraws when they change, like any dashboard. Nothing is rewritten again while it is open.

- **A binding is a code.** A Region's pickers store codes. Passed to `openPanel` or `openDashboard`, each code is written in quotes: `{PUMP_RUN: "P101_RUN"}`. Written bare, `{PUMP_RUN: P101_RUN}`, the variable's current value, such as `true`, would be pasted into the panel's formulas in place of its code.
- **What is left as written:**
  - text in double quotes, so a function that takes a variable's code as text, such as `history`, keeps the code written in the panel;
  - a name followed by `(`, which is a function;
  - a name after a `.`, which is a field of a record;
  - settings that are not formulas, such as a slider's Min and Max;

  - a Region's own bindings: see Panels inside panels.
- **Record keys are rewritten too.** A parameter named like a key that the panel writes in a record, such as `fill`, `color`, `label` or `value`, replaces that key in every record, and the style or option that used it stops working.
- **Name parameters for what they stand for,** in the plant's style: `PUMP_RUN`, `PUMP_SPEED`. Never name one like a style key, a record key or a variable of the plant: a parameter named like a plant variable replaces that variable everywhere in the panel once it is bound, and reads the plant variable while it is not.

Unbound parameters

A parameter the opener binds to no variable is left as written:

- the panel's formulas read a variable of the parameter's own name. When there is none, they fail and fall back, so the widget looks blank: see When a formula fails;
- an input bound to it writes a variable of that name, and creates it;
- the server log gets a warning that names the panel and its unbound parameters.

A panel can therefore open before every binding is filled in; a blank widget in a panel usually means a missing binding.

Closing

Opened by	Closes when
A Region **On click**	A click anywhere outside the panel. That click only closes the panel: it does not reach the widget under it
A Region **On hover**	The pointer has been off both the region and the panel for 250 ms. Moving onto the panel keeps it open, and coming back within the 250 ms keeps it too
`openPanel`	Only when `closePanel` names its id. The panel has no close control of its own: put an Action button with `closePanel` on it
Any of them	Another panel opens under the same id; the view that opened it closes or has its content replaced; or the panel that opened it closes

Panels fade in and out in 160 ms.

Opening panels from a script

The Site adds three functions that act only in an Action button's script. Anywhere else, they do nothing and give `null`.

Function	What it does
`openPanel(id, code, dX, dY [, bindings])`	Opens the panel `code` under `id`, with its centre `dX` and `dY` canvas pixels from the centre of the button that ran the script. `bindings` is a record of parameters and quoted codes. The id behaves as a Region's name: the same panel, bindings and place again changes nothing, and anything else replaces it. The panel closes only with `closePanel`
`closePanel(id)`	Closes the panel under `id`, wherever on the screen it was opened from, with every panel it opened. Does nothing when no panel is open under `id`
`openDashboard(code [, bindings])`	Replaces what the button's view shows with the dashboard `code`: the screen's content or, for a button on a panel, the panel's content, inside the panel's box. It closes the panels that view had opened. The tab keeps its title

- **Ids are shared on a screen,** between regions and scripts. `closePanel("PUMP_P101")` closes the panel that a Region named `PUMP_P101` opened, so a Close button on a panel closes it whatever opened it.
- **Bindings name declared parameters.** A key the panel does not declare is left out, with a warning in the server log, and a key with no code leaves its parameter unbound. A dashboard of category Dashboard declares no parameters, so the bindings of `openDashboard` matter only when its code names a panel.
- **A code that names nothing** opens nothing.

Example. A Close button on a panel opened by a script:

```
ActionButton  Label   "Close"
              Script  closePanel("pump_detail")
```

Example. A button in an alarm list that opens a pump's faceplate above itself:

```
ActionButton  Label   "P-101"
              Script  openPanel("pump_detail", "PUMP_FACEPLATE", 0, -140,
                                {PUMP_RUN: "P101_RUN", PUMP_SPEED: "SIT_101", PUMP_SPEED_SP: "SP_SIT_101"})
```

Panels inside panels

A panel can hold Regions and Action buttons that open further panels.

- **The inner panel is drawn in the screen's top layer,** placed from the region or button inside the outer panel, and is not cut off by the outer panel's box.
- **Closing a panel closes every panel it opened.**
- **A binding cannot be passed on:**
  - A Region's bindings are stored codes and are not rewritten. Bound to a parameter of the outer panel, the inner panel receives that parameter's name, and reads the variable of that name rather than the one the outer panel was bound to.
  - In an Action button's script, the keys of the bindings record are rewritten when they match an outer parameter, so the inner panel no longer recognises them and leaves them out; the quoted codes are never rewritten.

Bind inner panels to the plant's own variables, and give the parameters of every panel names that no other panel's parameters use.

Licence

Displaying dashboards and panels needs an edition that displays panels. On an edition that builds and edits panels but does not display them, no dashboard opens for viewing, and a Region opens nothing, with a warning in the server log. Building and editing dashboards and panels is never licensed. See How licensing works.

Example: one faceplate for every pump

The panel, written once:

```
Panel        PUMP_FACEPLATE   Category Panel   Width 320   Height 200
             Parameters       PUMP_RUN, PUMP_SPEED, PUMP_SPEED_SP
Circle       Styles    {fill: evl(if(PUMP_RUN, "#16A34A", "#9CA3AF"), "#9CA3AF"), stroke: "#374151", stroke_wid
Label        Label     evl("%.0f rpm".format(to_double(PUMP_SPEED)), "-- rpm")
FieldNumber  Variable  PUMP_SPEED_SP
```

A Region over each pump of the plant diagram:

```
Region PUMP_P101   Trigger On click   Panel PUMP_FACEPLATE   Popup dX 0   Popup dY -140
                   PUMP_RUN: P101_RUN   PUMP_SPEED: SIT_101   PUMP_SPEED_SP: SP_SIT_101
Region PUMP_P102   Trigger On click   Panel PUMP_FACEPLATE   Popup dX 0   Popup dY -140
                   PUMP_RUN: P102_RUN   PUMP_SPEED: SIT_102   PUMP_SPEED_SP: SP_SIT_102
```

A click on P-101 opens the panel with its centre 140 pixels above the pump's, drawn as if it had been written for P-101:

```
Circle       Styles    {fill: evl(if(P101_RUN, "#16A34A", "#9CA3AF"), "#9CA3AF"), stroke: "#374151", stroke_wid
Label        Label     evl("%.0f rpm".format(to_double(SIT_101)), "-- rpm")
FieldNumber  Variable  SP_SIT_101
```

A click outside the panel closes it, and does not reach the widget under it; a second click, on P-102, opens the same panel for the second pump.

Next steps

- Buttons
- Inputs
- Groups
- How dashboards draw and refresh

This page describes Data Orchester Dashboards 1.9.4.