# Board Admin Guide

A guide for board admins of the Fraznet internal dashboard at **dashboard.frazil.app**.

If you can edit a board, this is for you.

---

## Contents

---

## Concepts

- **Board** — a dashboard. Each board has its own widgets, groups, and filters. The backend still uses the word "team" in some places, but it means the same thing.
- **Group** — a section of a board. Widgets live inside groups. A group can be collapsed and can also hold filters.
- **Widget** — a single tile (stat, chart, table, map, etc.).
- **Snapshot** — the most recent value of every widget, cached on the server. The browser does not run your queries. The server precomputes them and the dashboard reads the cache.
- **Roles**
- `viewer` — read-only.
- `team_admin` — can edit boards they're a member of.
- `admin` — global. Can manage users, boards, and the admin tools.

---

## Signing in & switching boards

- The login screen asks for a username and password. The dashboard remembers your last board in your browser, so closing the tab and coming back drops you in the same place.
- The board name shows in the top bar. If you belong to more than one board, click it to switch.
- Global admins see every board in the switcher. Other users see only the boards they're members of.

### First sign-in (invitation email)

If an admin added you with the **Email invitation** option, you'll receive an email titled *"Welcome to the Fraznet Dashboard — set up your account"*. Click **Set your password** in the email, choose a password on the page that opens, then sign in normally with your username and the password you just picked. The invitation link is valid for 15 minutes; if it expires, ask the admin to resend it from your user record.

### Forgot password

1. On the sign-in screen, click **Forgot password?**.
2. Enter your username or your email address.
3. We send a reset link to the email on file (no message is shown about whether the account exists — that's intentional).
4. Open the email and click **Choose a new password**. The link works for 15 minutes.
5. Set a new password. You'll be redirected back to sign in.

Resetting your password signs out every active session for your account, including the device that triggered the reset.

---

## Your account

The chip at the top right of the dashboard shows your initials and username. Click it to open the user menu:

- **Admin** — opens the admin tools (visible only to global and board admins).
- **My account** — opens your account settings.
- **Sign out** — ends this session.

### My account modal

Three tabs:

**Profile** — your username and role are read-only. You can edit your **email** here. Only addresses ending in `@freezingpointllc.com` or `@frazil.com` are accepted. This is the address password-reset links are sent to.

**Password** — change your password. You'll need to enter your current password as a confirmation.

**Sessions** — every browser/device currently signed in to your account. The session you're using is marked **This device**; the others can be signed out individually.

---

## Edit mode

Most of the editing UI is hidden until you toggle **Customize / Edit**:

- A pencil/edit button in the top bar enters edit mode.
- The same button becomes **Done editing**; clicking it leaves edit mode and reloads the dashboard.

- Drag handles, add/edit/delete buttons, and "Add widget" drop zones only show in edit mode.

You don't need edit mode to use filters or change the polling interval — those are visible to everyone.

---

## Groups

Use groups to organize related widgets and to give yourself a place to collapse big sections.

### Adding a group

1. Enter edit mode.
2. Click **+ group**.
3. Pick a name, color, and (optional) page number.

### Group settings

| Setting | What it does |
|---|---|
| Name | Header label. |
| Color | Header accent + the default accent for widgets added inside it. |
| Page | Used by Presenter mode — groups with the same page number render together on a single slide. |
| Hide in kiosk | The group is skipped in Presenter mode. |
| Stack beside | Switches the group's layout to *column-dense* — useful for "narrow widgets stacked next to a tall map or chart" arrangements. |
| Widget title size | Overrides widget-title font size for every widget in this group. Default (11px) → XX-Large (24px). One group's setting doesn't affect other groups. Useful for big-TV presentations. |
| Collapsed | Saved per-user. The body stays hidden until you expand it, and any filter dropdowns hide with it. |

**Re-ordering** — drag the ■ handle on the left of the group header. You can also drag widgets between groups in edit mode.

**Copy group to another board** — click the ■ icon on the group header (edit mode, team_admin+) to clone a whole group (widgets, optionally filters) onto another board you have access to. Useful for templated layouts you want repeated.

**Board layout density** — Board settings ■ exposes a **Row height** picker (Tight 60 / Standard 100 / Generous 140 / custom). It's the *minimum* row height; widgets with denser content can grow above it.

---

## Group filters

A filter is a dropdown that lives in the group header. Pick a value and every widget in that group re-runs its query with that value substituted in.

Filters work via **placeholders** — you write a token like `{{country_filter}}` anywhere in a widget's query, and the filter rewrites the token into a SQL snippet before the query runs.

### Creating a filter

1. Edit mode → click ■ **filters** on the group header.
2. Click **+ Add filter** in the modal that opens.
3. Fill in:

| Field | Notes |
|---|---|
| **Name** | What users see next to the dropdown (e.g. "Country"). |
| **Placeholder** | The token to drop into queries, e.g. `country_filter` (becomes `{{country_filter}}`). |
| **Filter type** | **Dropdown** (default, single or multi-select) or **Date range** (two date pickers — see below). |
| **Template** | The SQL fragment that replaces the token. For dropdown filters use `{value}` where the picked value goes. Example: `t.country_id = {value}`. For date range filters use the date placeholders described below. |

| | |
|---|---|
| **Source SQL** | (Dropdown filters only.) A query that returns the dropdown options. Two columns: value and label. |
| **Value column** | (Dropdown only.) The column name (or position) holding the value passed back to your queries. |
| **Label column** | (Dropdown only.) The column name shown in the dropdown. |
| **Show "All"** | (Dropdown only.) Adds an "All" option that disables the filter for that picker. |
| **Multi-select** | (Dropdown only.) Picker becomes a checkbox list with Apply / Clear buttons. Template uses `{values}` (plural) inside `IN (...)`. |
| **Require selection** | Removes the "All" option. Widgets that reference this placeholder show a *"Pick X"* empty state until a value is chosen — nothing runs on the server until you pick. Use this to gate expensive widgets so they don't auto-run on board load. |

### Generic example

Source SQL for the dropdown:

```
SELECT id AS value, name AS label FROM countries ORDER BY name
```

Template: `t.country_id = {value}`

Widget SQL:

```
SELECT COUNT(*) AS orders FROM orders t WHERE t.created_at >= CURDATE() - INTERVAL 7 DAY
AND {{country_filter}}
```

When the user picks "Germany" (id `42`), the widget runs:

```
... AND (t.country_id = 42)
```

When the user picks "All" the placeholder becomes `(1=1)`, so the filter has no effect.

### Multi-select variant

When the filter has **Multi-select** enabled, the template uses `{values}` instead of `{value}` and goes inside an `IN (...)`. Example template:

```
c.chain_id IN ({values})
```

Picking three chains substitutes to `c.chain_id IN (12, 34, 56)`. With nothing selected it falls back to `(1=1)`.

### Date range filter

Set **Filter type** to **Date range** for a pair of native browser date pickers instead of a dropdown. The filter bar renders as `Start — End [Clear]`, and clicking either input pops up the browser's calendar.

The template uses four placeholders so a single template handles range / before-only / after-only:

| Placeholder | When filled by user | When left blank by user |
|---|---|---|
| `{start}` | the picked date | empty string |
| `{end}` | the picked date | empty string |
| `{start_or_min}` | the picked date | `1000-01-01` (sentinel) |
| `{end_or_max}` | the picked date | `9999-12-31` (sentinel) |

Recommended template that covers all four user behaviours in one line:

```
c.date >= '{start_or_min}' AND c.date <= '{end_or_max}'
```

| User picks | Resolved SQL | Meaning |
|---|---|---|
| Start `2026-01-01`, end `2026-06-30` | `c.date >= '2026-01-01' AND c.date <= '2026-06-30'` | Range |
| Start `2026-01-01`, end blank | `c.date >= '2026-01-01' AND c.date <= '9999-12-31'` | After 2026-01-01 |
| Start blank, end `2026-06-30` | `c.date >= '1000-01-01' AND c.date <= '2026-06-30'` | Before 2026-06-30 |
| Both blank (Clear) | substitutes to `(1=1)` — no filter | No filter |

The sentinels work for `DATE` and `DATETIME` columns. For numeric timestamp columns (e.g. Unix epoch) you'd want a custom template using `{start}` / `{end}` with explicit numeric defaults instead.

### Inverse variant — `{{name_not}}`

Every placeholder also exposes `{{name_not}}` which substitutes as `NOT (...)`. Useful for "everyone else" widgets next to a "yours" widget on a gamified board:

```
-- "Your team's count": AND {{team_filter}} -- "Everyone else's count": AND
{{team_filter_not}}
```

### The filter dropdown

- The picker is a searchable combobox. Click it, then type to filter the list.
- Arrow keys + Enter pick a value. Esc closes the panel.
- Filters reset to "All" on every page reload, so you never come back to a board with a stale filter applied.

### Things to know

- The same placeholder name can be referenced in the widget's main SQL, comparison query, and export SQL — all three get rewritten.
- A widget that doesn't reference the placeholder is unaffected by the dropdown. Only widgets that contain `{{...}}` are filtered.
- When a filter is active, snapshots are computed live (per-user) and are not cached. The dashboard recomputes only the widgets in groups with active filters; other groups serve from cache.
- While a filtered group is recomputing, the affected widgets dim and show a small spinner so you know the values are being refreshed.

### Removing or editing a filter

In the ■ **filters** modal, click any filter to edit it, or use the ✕ to delete.

---

## Widgets

Every widget shares these settings:

- **Title** — top of the tile.
- **Group** — which group it belongs to.
- **Display type** — what you're building (stat, chart, table, etc.).
- **Width** — ■ / ¼ / ■ / ½ / ¾ / Full (of the 12-column grid).

- **Height** — ½ / 1 / 2 / 3 / 4 rows. Use taller heights for maps and large charts.
- **Color** — accent.
- **Poll** — how often this specific widget re-runs (or "Global setting" to follow the board-wide timer).
- **Alignment**, **Delta**, **Value format**, **Timezone** — shown for the widget types that use them.
- **Export** (collapsible) — optional SQL/script that produces a CSV download.

Each query box has a small **SQL / R / Python** toggle. SQL is the default; the other two run a script on the server.

The widget editor's **Cancel** and **Save widget** buttons stay pinned to the bottom of the modal as you scroll, so you don't lose them while editing a long query.

### Stat

The "big number" tile.

Query shape: returns exactly **1 row, 1 column**.

```
SELECT COUNT(*) FROM orders WHERE status = 'paid'
```

Use **Value format** to format as currency, percent, thousands, etc. Use **Delta** to compare against either the previous poll or a second query.

**Value scaling** controls how aggressively short numbers grow to fill the card. The dashboard always auto-fits the number to the card width — long values shrink, short values grow up to the cap.

| Setting | Cap (multiple of the column's baseline font) | When to use |
|---|---|---|
| Auto / 1.8× *(default)* | grows up to 1.8× baseline | Good fit for most boards. |
| Off (1.0×) | no growth (CSS baseline only) | When you want every stat to be the same size for visual uniformity. |
| 1.2× / 1.5× | mild growth | Compromise between uniform and "fill the card". |
| 2.0× / 2.5× / 3.0× | aggressive growth | Big-TV presentations where you want the number to dominate. |

Per-widget settings are respected in both dashboard and Presenter mode, so a widget set to 2.5× will scale up to 2.5× the col-based baseline in both views.

### Gauge

A stat with a circular fill showing how close you are to a target.

Query shape: 1 row, 1 column (the current value). Set the max in the gauge config.

```
SELECT SUM(amount) FROM payments WHERE day = CURDATE()
```

### Status indicator

A colored-dot stat that flips between **good / warn / bad** based on thresholds.

Query shape: 1 row, 1 column (the current value).

```
SELECT (open_count * 100.0 / total_count) AS pct_open FROM ticket_stats
```

Configure:

- **Direction** — *Higher is better* (default) or *Lower is better*.
- **Good threshold** — value at/past this is green.
- **Warn threshold** — value at/past this is amber. Anything beyond is red.
- **Custom labels** for the three states (e.g. "Healthy" / "Watching" / "Critical").

Empty thresholds disable that state — leave **Warn** blank if you only want green/red.

### Line chart

A stat with a small line chart showing recent points (collected from each poll).

Query shape: 1 row, 1 column. Each poll appends one point to the trailing history. Pick a timezone for the X-axis labels.

### Bar chart (time series)

Same as line chart but rendered as bars. Use **Orientation** to switch between horizontal and vertical.

### Progress

A stack of labeled progress bars.

Query shape: one query per row, pipe-delimited:

```
Label 1 | SELECT COUNT(*) FROM things WHERE x = 'a' Label 2 | SELECT COUNT(*) FROM things WHERE x = 'b' Label 3 | SELECT COUNT(*) FROM things WHERE x = 'c'
```

Set the **Max** as a constant or as its own SQL query. Each bar fills to `value / max`.

## Badges

A grid of label/value cards.

Same pipe-delimited shape as Progress:

```
Active users | SELECT COUNT(*) FROM users WHERE last_seen > NOW() - INTERVAL 1 DAY Signups today | SELECT COUNT(*) FROM users WHERE DATE(created_at) = CURDATE()
```

Each value can carry its own format if you add a third `|` field:

```
Revenue | SELECT SUM(amount) FROM payments | currency
```

**Section headers.** Lines starting with `---` become full-width section headers that group the badges below them. Each section has its own auto-fit grid, so a section with one badge stretches to fill the whole row:

```
--- This Week Orders | SELECT COUNT(*) FROM orders WHERE created_at >= DATE_SUB(CURDATE(), INTERVAL 7 DAY) | integer Revenue | SELECT SUM(total) FROM orders WHERE created_at >= DATE_SUB(CURDATE(), INTERVAL 7 DAY) | currency --- Notes Open issue | SELECT title FROM issues WHERE status='open' ORDER BY priority LIMIT 1
```

**Layout dropdown.** In the widget editor, **Layout** controls how badges share horizontal space:

- **Equal-width grid (compact ~140px min)** — default. Equal columns, fits ~3-4 across a `col-3` widget. Best for many small KPIs.
- **Equal-width grid (wide ~220px min)** — same idea but each badge is at least 220px wide. Best for hero stats on `col-6`+ widgets.
- **Flow — size to content (legacy)** — chips size to their content. Ragged columns but values never get cramped.

**Long values.** Long label or value text can either wrap to multiple lines (default) or **Truncate with ellipsis + hover tooltip**. The truncate mode is what you want when a cell holds something like a store name or address that might overflow.

## Cache table

Like Badges, but shows a small Δ next to each value compared to the previous poll. Useful as a "what changed" board.

## R Script

Run an R script and display a single value (or pull richer output via the language toggle on other widgets).

Script must `print(...)` or `cat(...)` a single value. Example:

```
data <- c(1, 4, 6, 8) print(mean(data))
```

Default script timeout is 30 seconds.

## HubSpot widget

Every query field has a HubSpot language option in addition to SQL / R / Python. When you pick **HubSpot**, the textarea becomes a Python snippet that runs server-side with `HUBSPOT_TOKEN` already set as an env var — so you can write requests against the HubSpot v3 API without dealing with auth.

Two ways to author HubSpot widgets:

- **Library picker** — Admin → HubSpot keeps a library of named snippets ("Open deals by stage", "Tickets created this week", etc.). On a widget editor, pick a name from the dropdown to fill the textarea.
- **No-code builder** — in the library editor, the "Build" panel takes object type (Deals / Contacts / Companies / Tickets / Line items), aggregation, group-by, filters, optional time window, sort, and limit. Compile to a snippet and save it to the library.

Output contract matches the widget type, same as R/Python.

## Category bars (multibar)

A bar chart driven by a 2-column query: label, value. **Optional third column** sets the per-bar color (see Per-bar / per-slice colors from SQL).

```
SELECT category AS label, SUM(amount) AS value FROM transactions GROUP BY category ORDER BY value DESC LIMIT 10
```

Use **Orientation** to flip horizontal/vertical and **Sort** to control ordering. Sort options include **Query order** — choose this when your SQL has its own `ORDER BY` (e.g. numeric bin labels that wouldn't sort right alphabetically).

## Grouped chart

A grouped/stacked bar chart driven by a 3-column query: group, series, value.

```
SELECT region AS series, month AS label, SUM(amount) AS value FROM sales GROUP BY region, month ORDER BY month
```

Each unique `series` becomes a colored set of bars. The **Style** picker switches between Bars (default), Stacked bars, Line, and Area.

## Funnel

A funnel chart — horizontal bars centered and tapering by value.

Query shape: 2 columns (label, value). Row order is the funnel order, top to bottom:

```
SELECT stage AS label, COUNT(*) AS value FROM leads GROUP BY stage ORDER BY stage_order
```

The widest stage is whichever the query returns first. Inline values can be hidden via the eye-icon labels toggle.

## Histogram

A continuous bar chart for distributions. Server bins raw observations into equal-width buckets and renders the per-bucket counts.

Query shape: **one numeric column** — one row per observation.

```
SELECT TIMESTAMPDIFF(HOUR, created_at, NOW()) AS hours_open FROM tickets WHERE status = 'OPEN'
```

Set **Bin count** (2–100) in the widget config. The server computes `[min, max]`, divides it into N equal-width buckets, and counts.

**Custom-binned histogram via Category bars** — for non-equal-width bins (e.g. "0–1 is one bucket, then 2-wide bins, then a 60+ catchall"), use Category bars with `CASE … END` to bin in SQL and the optional 3rd column for per-bar colors.

## Calendar heatmap

A grid of cells colored by intensity. Two layouts:

- **Day-of-week × hour-of-day** — 7×24 grid. Query returns `(dow, hour, value)` where `dow` follows MySQL's `DAYOFWEEK` convention (1=Sun … 7=Sat).
- **Calendar (last 365 days)** — query returns `(date, value)`, one row per day.

Color intensity scales from min to max within the dataset.

## Result table

A pageable, sortable table. Accepts any multi-column SELECT.

```
SELECT order_id, customer_name, amount, created_at FROM orders ORDER BY created_at DESC LIMIT 200
```

Set **Max rows** in the widget config to cap how many are fetched. Users can click column headers to sort and use the pager controls at the bottom.

**Highlight rules** — color cells based on conditions. One rule per line:

```
column OP value -> color
```

Operators: `=`, `!=`, `>`, `>=`, `<`, `<=`, `contains`. Rules are evaluated top-down, first match wins per cell.

Colors accept either palette keys (`red`, `amber`, `green`, `frazil`, `teal`, `slate`, `pink`, `orange`) **or CSS hex codes** (`#86efac`, `#0057B8`, `#f8a`). Text contrast is auto-picked by luminance, so you can use any background without worrying about readability.

```
store_cpm < 2 -> #e8394a store_cpm < 12 -> #ff8a95 store_cpm < 18 -> #86efac store_cpm < 28 -> #22b96e store_cpm < 36 -> #15803d store_cpm >= 36 -> #0057B8
```

## Pie / Donut

A 2-column query: label, value. **Optional third column** sets the per-slice color (see Per-bar / per-slice colors from SQL).

```
SELECT plan AS label, COUNT(*) AS value FROM subscriptions GROUP BY plan
```

Pick **Style** (pie vs donut) and **Legend** placement in the widget config. Inline slice labels show value + percentage for slices ≥ 4% (toggleable via the eye-icon in the widget header).

## Map

A Leaflet map. Available geocode modes:

| Geocode mode | Query shape | Notes |
|---|---|---|
| `lat_lng` | `lat, lng, label, value` | Markers at exact coordinates. |
| `us_state_centroid` | `state, label, value` | State name or 2-letter abbreviation. Marker at the state's centroid. |

| | | |
|---|---|---|
| `us_state_choropleth` | `state, value` | Fills each US state by value (light → saturated using the widget color). |
| `ca_province_centroid` | `province, label, value` | Canadian provinces. Name or 2-letter code (AB, BC, ON, QC, …). |
| `ca_province_choropleth` | `province, value` | Filled Canadian provinces. |
| `na_centroid` | `region, label, value` | Mixed US-state and CA-province codes in the same result. |
| `na_choropleth` | `region, value` | Filled across both countries — single value scale. |
| `service_region_choropleth` | `region_id, region, value` | Internal service regions (zip-aggregated). One row per region. Region polygons are pre-merged from the zip mapping. |

For state / province / region modes, the centroid label shows the value (and region name) by default — the eye-icon toggle in the widget header hides them if you just want the colored fill.

For `service_region_choropleth`, region IDs disambiguate duplicate names (e.g. two "Huntsville" regions in different districts). Querying with `region_id` (rather than just the name) ensures each region matches its own polygon. `value` can be `NULL` to render coverage-only — colored regions, no number labels.

Map widgets look best at **2- or 3-row height**.

## API widget

Read JSON from an external HTTP endpoint and display a field from it.

Fields:

- **URL** — required.
- **Method** — `GET` (default), `POST`, etc.
- **Headers** — JSON object, e.g. `{"Authorization":"Bearer ..."}`.
- **Body** — for POST.
- **JSON path** — dotted/bracketed path to the value, e.g. `data.results[0].count`.
- **Display** — `stat`, `badges`, `table`, or `chart`.

### Note (Markdown)

A static markdown panel for notes, links, or context next to your widgets.

Supports standard markdown — headings, lists, bold/italic, links, code blocks, blockquotes.

---

## Widget sizing, height and color

### Width

Sets how many of the 12 grid columns the widget spans.

| Width | Columns | Visual |
|---|---|---|
| ■ | 2 | tiny stat |
| ¼ | 3 | default for stats |
| ■ | 4 | medium |
| ½ | 6 | half-row |
| ¾ | 9 | wide |
| Full | 12 | full row |

### Height

Sets how many rows the widget spans vertically.

| Height | Best for |
|---|---|
| ½ row | Compact stats |
| 1 row (default) | Most things |
| 2 rows | Charts, result tables, multi-state badges |

| | |
|---|---|
| 3 rows | Maps, big bar charts |
| 4 rows | Detailed result tables, choropleths with leader labels |

### Color

Pick from the accent palette: frazil, accent, green, red, amber, teal, pink, orange, indigo, slate, brown, yellow, black. The same palette also renders well in the dark theme used by Presenter mode.

---

## Value formats

Most stat/chart widgets format their value via the **Value format** picker:

| Format | Output example |
|---|---|
| auto | 1,234 / 1.2M |
| integer | 1,234 |
| double / float | 12.34 |
| currency (USD) | $1,234.56 |
| currency_gbp | £1,234.56 |
| currency_eur | €1.234,56 |
| percent | 12.3% (input already a percent) |
| percent_raw | 12.3% (input is `0.123`) |
| thousands | 1.2K |
| millions | 1.2M |
| duration_s | 1h 23m |
| text | shown as-is |

## SQL / R / Python

Every query box has a **SQL / R / Python** toggle. Pick the language the source is in.

### SQL

Read-only queries against the production replica. The following are allowed: `SELECT`, `SHOW`, `WITH` (CTEs). The following are blocked and will fail: `INSERT`, `UPDATE`, `DELETE`, `DROP`, `CREATE`, `ALTER`, `TRUNCATE`, `GRANT`, `REVOKE`, `CALL`, `EXECUTE`, `LOAD`, `OUTFILE`, `DUMPFILE`.

The replica is read-only so you can't write even if the validator missed something. Still, please write your queries with a `LIMIT` where it makes sense — the server caps result rows at 10,000.

### R

Scripts run via Rscript on the server with a 30-second timeout. Print/cat your output. Output contract by widget type:

| Widget | Output |
|---|---|
| stat / line / bar / gauge | `print(42)` |
| badges / progress / cache table | `cat('{"Label A":10,"Label B":20}')` (progress adds `"__max":100`) |
| multibar / pie / map | `cat('[{"label":"A","value":10},{"label":"B","value":20` |
| resulttable | `cat('[{"col1":1,"col2":"x"}, ...]')` |
| export | `cat('[{"col1":1,...}, ...]')` |

### Python

Same model as R. `print(...)` your output. Same widget output contracts.

R and Python are great for any computation that's awkward in SQL: rolling windows, ad hoc joins from API data, anything statistical. They can also be used to fan out to multiple data sources, since the script can issue its own HTTP/DB calls.

## Comparison (delta) queries

Stat-style widgets (stat, line, bar, R script, gauge) show a small Δ indicator. Choose how it's calculated:

| Delta mode | What it does |
|---|---|
| Off | Hide the indicator. |
| Previous value | Δ vs the value from the previous poll. |
| Comparison query | Δ vs the result of a second query (e.g. "same period last week"). |

A comparison query can be SQL / R / Python — it has the same toggle as the main query.

Generic example — "vs last week":

```
SELECT COUNT(*) FROM orders WHERE created_at >= NOW() - INTERVAL 14 DAY AND created_at < NOW() - INTERVAL 7 DAY
```

---

## Per-bar / per-slice colors from SQL

**Category bars** and **Pie / Donut** accept an **optional third column** in the query that drives the per-element color, overriding the series-color picker. Useful when each bar / slice should reflect a semantic state (status, threshold band, geography, etc.) that's easier expressed in SQL than configured by hand.

```
SELECT status AS label, COUNT(*) AS value, CASE status WHEN 'OPEN' THEN '#e8394a' WHEN 'DISPATCHED' THEN '#f59e0b' WHEN 'CLOSED' THEN '#22b96e' WHEN 'CANCELED' THEN '#475569' ELSE 'frazil' END AS color FROM tickets GROUP BY status
```

- Accepts either palette names (`red`, `frazil`, `green`, …) or CSS hex codes (`#86efac`, `#0057B8`).
- Per-row `NULL` falls back to the series picker for that index — mix freely.
- Inline labels' text color is auto-picked by luminance, so hex backgrounds stay readable.

---

## Inline labels toggle (eye icon)

Some widget types render decorative labels (region names on maps, value labels on charts) that take up space. Hide them per-widget with the **eye-icon** in the widget header.

Supported widgets: **Map**, **Pie / Donut**, **Category bars**, **Funnel**.

- Click the eye icon to toggle. The choice is saved to the widget config — all viewers see the same state.
- Same toggle is available as a **Show labels** checkbox in the widget editor.
- Tooltips and the colored fill always stay — only the always-on text labels hide.
- The setting is respected in Presenter mode (kiosk) too.

---

## Drill-down clicks

For widgets where individual elements are clickable (a bar, a slice, a heatmap cell, a table row), set a **Drill-down filter target** in the widget editor. Clicking an element on the source widget applies the clicked label as a temporary filter on the named placeholder.

Useful for "click a region on the map → the rest of the group narrows to that region's stores" patterns. Set it up by:

1. Have a group filter with placeholder `region` (for example) on the group.
2. On the source widget, set **Drill-down target** to `region`.
3. Click a bar/slice on that widget — the filter dropdown switches to that value and the rest of the group reloads with it applied.

Click the same element again (or pick **All** on the filter) to clear.

---

## Chart annotations

**Category bars** and **Grouped chart** support overlay annotations — horizontal target lines, vertical guidelines, or labeled markers on specific bars.

One rule per line in the **Annotations** textarea:

```
line | y=20 | red | Target line | x=Oct | amber | Holiday freeze starts
```

Format: `line | (y=value | x=label) | color | optional label`.

Skip the section entirely if you don't want any annotations.

---

## PNG export

Every widget has a small **PNG download** icon in the header that snapshots the rendered widget (including chart axes, colors, and labels) as a `.png` file. Useful for slide decks or quick screenshots without cropping.

The export captures the *current* visual state — applied filters, eye-icon label toggle, etc. all reflected.

## Export to CSV

Every widget can have an **Export** query (collapsible section in the editor). When users click the ■ icon on the widget, the export query runs and downloads a CSV.

- The query can be SQL, R, or Python (with its own language toggle).
- It can return more columns than the widget itself shows — handy for "the chart shows top 10 by revenue, the export gives me the full row with timestamps".
- Group filters apply to the export query the same way they apply to the main query.

---

## Threshold alerts

Single-value widgets (stat, gauge, line, bar) can fire alerts when their value crosses a threshold. Alerts deliver to a notification channel (Slack incoming webhook or email).

**Setup is in two parts:**

1. **Admin → Alerts → Notification channels** — global admins add Slack and/or email channels. Each channel has a name, a target (webhook URL or email address), and a **Test** button that sends a sample message.
2. **Per-widget alert rule** — on the widget itself (a bell icon in the header for team_admin+), open the rules modal. Add one or more rules: `<operator> <threshold> → <channel>`, e.g. `>= 100 → ops-alerts`.

Rules fire only on **edge crossings** (ok → firing). Once a rule is firing it won't re-fire until the value goes back to ok. The bell badge on the widget header turns red while a rule is firing.

The most recent 100 firings show in **Admin → Alerts → Recent firings**, including delivery status.

---

## Saved queries & R script library

The board has two reusable libraries you can pull from while editing widgets:

- **Saved queries** — named SQL snippets.
- **R scripts** — named R scripts (and Python, despite the name).

Save anything you reuse — common joins, "rolling 30 days" patterns, expensive subqueries — and you'll find them in a dropdown above the query box when editing widgets.

---

# Presenter mode (kiosk)

Use Presenter mode for displays on TVs or in meeting rooms. It loops through a list of slides full-screen with a dark theme.

### Setting up

1. Click the screen-icon button in the top bar → **Presenter setup**.
2. Build a slide list: each entry is either a **board** (renders that board's groups), a **HubSpot dashboard** (registered under Admin → HubSpot → Dashboards), or a **URL** (renders any web page in an iframe).
3. The **Add board** dropdown groups its options into **Boards** and **HubSpot dashboards** — HubSpot entries show a `(HS)` suffix so they're easy to spot.
4. Drag to reorder. The order is per-user and saved.
5. Pick a **cycle interval** (how long each slide is shown).
6. Optionally enable **shuffle**.
7. Optionally enable **keep screen awake** — uses the browser's Wake Lock API.
8. Click **Start**.

**Pagination on a single board** — groups with the same **Page** number render together; groups on different pages become separate slides. So you can split one board into multiple presenter slides without making multiple boards.

**Hide from kiosk** — flag set on a group hides it entirely from the presenter loop while still showing it on the editable dashboard.

**Exit** — Esc, or click the ✕.

## External dashboards (HubSpot, Genesys, etc.) in presenter mode

Vendor dashboards typically refuse to render in an iframe without their own session cookie, so unattended kiosks can't just point at the raw URL. The dashboard solves this with a server-side **PNG snapper** (`snapshot_runner.py`) that screenshots each registered external dashboard hourly and serves the cached image to the kiosk — no browser session needed on the TV.

The system supports multiple **kinds** of external sources:

- **HubSpot** — registered via the legacy HubSpot dashboards card. URL parsed automatically.

- **Genesys** — added via the "Other external dashboards" card with a kind dropdown.
- **Salesforce / Generic** — anything else, same flow as Genesys.

### Registering an external dashboard for kiosk use

1. Admin → **External**.
2. For HubSpot: **HubSpot dashboards** card → **+ Add dashboard**, paste the `/reports-dashboard/...` URL.
3. For Genesys / Salesforce / Generic: **Other external dashboards** card → **+ Add source**, pick the kind, paste the URL, set the session-file path (auto-defaults from kind), optionally set a capture hook.
4. **Seed the session** (one-time per source). On your laptop run: `python bootstrap_session.py --source <source_id> --url <URL> --out <local-path-to-session.json>` Firefox opens; complete the login (SSO, 2FA, whatever the vendor requires). Press Enter when you're inside the dashboard. The script saves the session JSON and prints the `scp` command to deploy it to the server's `session_file` path.
5. Click **Snap now** in the admin table to verify. The PNG appears in `/static/snapshots/{source_id}.png` and the row's status flips to `✓ ok`.
6. In **Presenter setup**, the source appears in the **Add board** picker under "External dashboards" with a kind suffix (`(HS)`, `(GEN)`, etc.). Pick it, click + Add.

### What the snapper does automatically

| Step | What | Why |
|---|---|---|
| Activates HubSpot **Full screen view** (when `capture_hook=hubspot_fullscreen`) | Hides HubSpot's nav, sidebar, breadcrumb chrome | Screenshot is just the dashboard |
| Dismisses Genesys toasts + hides orphan tab strip (when `capture_hook=genesys_dismiss_toasts`) | Removes "Select a phone" banner and leftover queue tabs | Clean kiosk display |
| Optional dark-mode transform per source | invert + hue-rotate 180° + gamma 0.80 + saturation 1.30 + contrast 1.30 | Matches the kiosk's dark theme; chart colors stay roughly correct |
| Stitches multi-page dashboards (HubSpot, sometimes others) | Scrolls + slices at widget boundaries via DOM detection | Tall dashboards get split into N pages with no chopped widgets |
| Waits for visual stability between scrolls | Polls thumbnail screenshots until two frames match for 700ms | Charts that animate in don't get screenshot mid-render |

### Session expiry — the only thing you'll need to repeat

Vendor session cookies last anywhere from a week (Genesys/Salesforce) to a month (HubSpot). When the snapper sees `Redirected to login (session invalid)` or `failed` in the row's status, that's the cue to re-bootstrap that source:

1. Run `python bootstrap_session.py --source <source_id> --url <URL> --out <local.json>` on your laptop again.
2. Sign in (SSO + 2FA as needed). Press Enter.
3. Run the printed scp/ssh command to push the new session JSON to the server.
4. Click **Snap now** in the admin row to verify the next snap works.

### Assigning external dashboards to specific users

Like boards, external dashboards can be **viewer-only assignments** per user. Admin → Users → Edit → scroll to **External dashboards** section → pick from the dropdown → + Add. The user sees the source in their board switcher dropdown under an "EXTERNAL" subheading with a kind badge (HS / GEN / etc.). Clicking it switches to a read-only view that stacks all the dashboard's pages.

---

## Install as an app (PWA)

The dashboard is a Progressive Web App — you can install it as a desktop app or home-screen icon.

**Why bother?** No browser chrome (URL bar, tabs), a dedicated taskbar/Dock icon, faster cold start, and it alt-tabs as "Fraznet" rather than "Chrome — Fraznet Dashboard".

### To install:

- **Chrome / Edge (desktop)** — click the install icon in the address bar, or ■ menu → "Install Fraznet…" / "Apps → Install this site as an app."
- **iOS Safari** — Share → "Add to Home Screen."
- **Android Chrome** — ■ menu → "Add to Home screen" / "Install app."

**To uninstall** — open the app window, click … menu → "Uninstall Fraznet…". Or on Windows, Settings → Apps. Your data and login stay intact across install/uninstall.

The PWA install doesn't make the dashboard offline — data still needs a network connection. It's mostly a convenience win.

---

## Refresh model & polling

The dashboard does not run your queries in the browser. The flow is:

1. The server stores a **snapshot** — one entry per widget — in SQLite.
2. The browser asks the server to refresh the snapshot.
3. The server respects each board's **Min recompute interval** (default 300s) and returns the cached snapshot if it's still fresh.
4. The server returns the snapshot to the browser, which renders it.

You'll see this in two places:

- **Global poll interval** (top bar) — how often the whole snapshot is refreshed. Defaults to 30 minutes. "Off" stops polling.
- **Per-widget poll** — set on individual widgets. Use for things you want to refresh faster than the rest of the board.
- **Manual refresh only** — set the widget's **Poll** to `Manual` (`-1`). The widget is **skipped** on every bulk board refresh, so a slow or flaky widget (e.g. an external API call that occasionally hangs) doesn't make the whole board's refresh wait for it. The widget keeps showing its last successful value until it's recomputed individually. Use this for HubSpot widgets, anything calling third-party APIs, or any single widget known to take too long.

The **Min recompute interval** is a per-board safety net (Board settings ■). It prevents users from hammering the database — even if the browser asks every 5 seconds, the server will still serve the cache until the interval has elapsed. Bump it up for boards that hit expensive queries.

When a group filter is active, the cache is bypassed for that group only — the filtered values are recomputed live per-user, and other groups still serve from cache. Repeated picks of the **same** filter value within a few minutes hit an in-memory result cache, so flipping back and forth between selections is near-instant.

---

## Query timeout

Every SQL query the dashboard runs against the production replica has a server-side timeout. If a query exceeds it, MySQL kills the query and the widget shows an error.

**Default** — 60 seconds (set by the `MYSQL_QUERY_TIMEOUT` env var on the server).

**Global override** — Admin → System has a **Default query timeout** input (5–600 seconds). Set this if your boards regularly run queries that need more than 60s, but you'd rather not bump every widget individually.

**Per-widget override** (global admin only) — the widget editor has a **Query timeout override** field below the Poll interval. `0` = use the system default. Set a higher value for a single heavy widget without changing the global setting.

Resolution order: per-widget override (if > 0) → system setting (if set) → env default. Whichever applies is the one MySQL enforces on that widget's query.

If a widget keeps timing out even after raising its override, the query itself probably needs work (missing index, unbounded join, etc.) rather than more time.

---

# Admin tools (global admin)

Open the admin tools by clicking the **user chip** → **Admin** in the top right. The main Monitoring and Queries tabs hide while you're in the admin view, and a ← **Back to dashboard** button at the top brings you back when you're done.

The admin view has a left sidebar with these sections (visibility depends on your role):

- **Users** — always visible.
- **Boards** — global admin only.
- **External** — global admin only. Includes HubSpot dashboards plus Genesys / Salesforce / generic external sources.
- **Alerts** — global admin only.
- **System** — global admin only.

## Users

The user table shows username, email, role, board, and last login. From here you can:

- **+ Add user** — create a new account (see below).
- **Edit** — change role/board/email, set a new password, or send a password-reset email.
- **Remove** — delete the account (you can't remove yourself).

### Adding a user

The **Add user** modal asks for:

- **Username**.
- **Email** — must end in `@freezingpointllc.com` or `@frazil.com`. (We enforce this on both the client and the server. The dashboard is for internal accounts only.)
- **Password setup** — pick one:
- **Set password now** — you type the initial password, the user signs in with whatever you give them.

- **Email invitation to set their own** — the user receives a welcome email with a 15-minute link to choose their own password. Email is required for this option. This is the recommended path for new hires.
- **Role** and **Board**.

#### Editing a user

The **Edit user** modal lets a global admin change the user's role, primary board, additional board memberships, email, and password. Global admins also see a ✉ **Send password reset email** button — useful when someone is locked out. The reset link is sent to the email on file and is valid for 15 minutes.

## Boards (global admins only)

Create, rename, configure, and delete boards. Per-board settings:

- **Min recompute interval** — cache freshness window for the board's snapshot (default 300s; bump for expensive boards).
- **Row height** — minimum widget row height (Tight 60 / Standard 100 / Generous 140 / custom). Widgets can grow above this; they won't compress below.

## External (global admins only)

This page combines the HubSpot integration with the broader external-dashboard snapper system.

- **HubSpot API connection** — paste/save the HubSpot Service Key (or legacy Private App token). Widgets running Python see it as the `HUBSPOT_TOKEN` env var.
- **HubSpot query library** — named Python snippets that widgets can reuse via the HubSpot language picker. Each entry has a ■ **Test snippet** button that runs it live and shows the output, plus a **Strip indent** button for cleaning pasted code. The library also has a no-code form builder that compiles a snippet from object type / aggregation / filters / etc.
- **HubSpot dashboards (PNG snapshots)** — list of HubSpot dashboard URLs the snapper screenshots on a timer. Click **+ Add dashboard** to register a new one (paste the `/reports-dashboard/...` URL).
- **Other external dashboards (Genesys, Salesforce, generic)** — same idea, different kinds. Click **+ Add source** to wire one up; the modal asks for kind, label, URL, session file path, optional capture hook, and dark-mode toggle.

Each row in either dashboards table exposes:

- **Last snapshot** — colored status pill + relative time. Hover the ■ failed pill to see the error.
- **Pages** — how many pages the snapper sliced this dashboard into.
- **Dark mode** — toggleable checkbox. Affects future snaps only.

- **Kiosk URL** — Copy URL button puts the wrapper URL on your clipboard (`/snapshot/<id>` or `/hubspot/<id>`).
- **Snap now / Remove** — manual recapture or remove the source.

### Alerts (global admins only)

- **Notification channels** — Slack incoming webhooks and email addresses. Each has a **Test** button.
- **Per-widget rules** are configured *on the widget* (bell icon), not here — but you'll see the full list across all widgets here for auditing.
- **Recent firings** — last 100 alert firings, including delivery success/error.

### System (global admins only)

- **Default query timeout** — global setting for how long a single MySQL query can run before the server kills it (5–600 seconds). See Query timeout.
- **Server caches** — manual flush buttons for the two server-side caches that speed up repeated work:
- **Filtered-snap cache** — caches widget results per filter combination so repeated filter clicks are instant. Flush after editing a filter template or fixing a widget SQL.
- **CPM eligible cache** — caches the latest-row-per-store scan of `fraznetapp_cpm` for the `{{cpm_eligible_stores}}` / `{{cpm_latest}}` SQL tokens (60-second TTL). Flush rarely needed; mostly used when debugging.

---

## Troubleshooting

#### "Loading…" forever on a widget

- The query may have failed. Hover the widget — an error icon appears with the message.
- The query may be returning the wrong shape. Check the **Query shape** column in the Widgets table for what each type expects.
- For R/Python widgets, check that you `print()`/`cat()` exactly what the widget type expects.

#### Error icon with a SQL message

- The most common cause is a banned keyword. See SQL / R / Python.
- "Snapshot too old / N/A" — when a filter is active and your widget doesn't reference the filter's placeholder, the widget is marked N/A on purpose.

### Filter dropdown is empty

- The **Source SQL** for the filter may have returned no rows. Edit the filter and re-test.
- The source SQL must include both a value and label column matching the names you set in **Value column** / **Label column**.

### Map shows nothing

- Check that `lat`/`lng` are valid numbers (and in that column order).
- For state maps, confirm you're returning either a state name or a 2-letter abbreviation (case-insensitive).
- Try increasing the widget's height to 2 or 3 rows — Leaflet won't render in a near-zero-height container.

### Charts look identical across reloads

- Trailing history (line/bar) is cached locally in your browser per-widget. New points are appended at each poll. If you change the query, clear the chart cache or change the widget ID by deleting and re-creating.

### Edit/delete buttons missing

- You're in viewer mode for that board. Ask a global admin to elevate your role on that board.

### Status badge is red ("Error") and won't go green

- The most recent snapshot refresh failed. Common causes: a widget query is broken, a per-widget timeout, or the DB pool is exhausted. Reload the page; if it persists, check with a global admin.

### I forgot my password / I can't sign in

- Use **Forgot password?** on the sign-in screen and enter your username or email. A reset link will be sent if the account has an email on file. (If you don't get an email after a minute, an admin can resend the link from your user record.)
- The reset link expires after 15 minutes — request a new one if you missed the window.

### The invitation email never arrived

- Check your spam folder for a message from `notifications@frazil.app`.
- Confirm with the admin that they used your `@freezingpointllc.com` or `@frazil.com` address.
- The admin can resend the invitation from the **Edit user** modal with ✉ **Send password reset email**.

### The dashboard works in light mode but not in Presenter mode

- Map widgets, charts, and color choices auto-switch to a dark variant in Presenter mode. If a custom HTML/markdown note looks wrong, set its colors with CSS variables (`var(--frazil)`, `var(--text)`, etc.) rather than hard-coded hex values.

# Quick reference

## Roles

| Role | Can do |
|---|---|
| viewer | View boards you're a member of. |
| team_admin | Edit boards you're a member of. |
| admin | Manage users, boards, and global settings. |

## Widget query shapes

| Widget | Expected shape |
|---|---|
| stat / line / bar / gauge / status | 1 row, 1 column |
| badges / progress / cache table | `Label \| SELECT ...` per line |
| multibar | label, value [, color] |
| pie / donut | label, value [, color] |
| funnel | label, value |
| histogram | one numeric column (raw observations) |
| grouped chart | series, label, value |
| heatmap (dow_hour) | dow, hour, value |
| heatmap (calendar) | date, value |
| resulttable | any multi-column select |

| | |
|---|---|
| map (lat_lng) | lat, lng, label, value |
| map (state / province / na centroid) | region, label, value |
| map (choropleth) | region, value |
| map (service_region_choropleth) | region_id, region, value |

### Sizes

| Width | Cols (of 12) |
|---|---|
| ■ ¼ ■ ½ ¾ Full | 2 3 4 6 9 12 |

### Account essentials

- Email domains accepted: `@freezingpointllc.com`, `@frazil.com`.
- Reset / invitation links expire after 15 minutes.
- Reset emails come from `notifications@frazil.app`.

**Min recompute interval** — per-board, defaults to 300s. Bump for expensive boards.

---

Questions, bug reports, or feature requests: ping `jr.frisby@frazil.com`.