Route Killer (beta) — Problem Solve Squad Guide

Audience: Amazon Problem Solve Squad

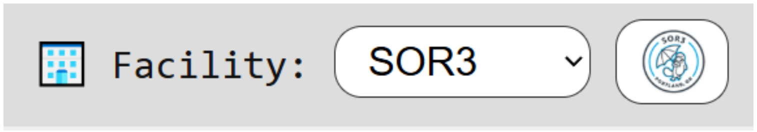
Purpose: Quickly translate outbound **routes** (two-letter codes) into **floor locations** on a facility map for route chasing, highlight them on an interactive layout, and share selections.

> 📸 **Screenshots:** This guide marks places where screenshots help. Please capture and insert your own screenshots in those spots.

UI Tour

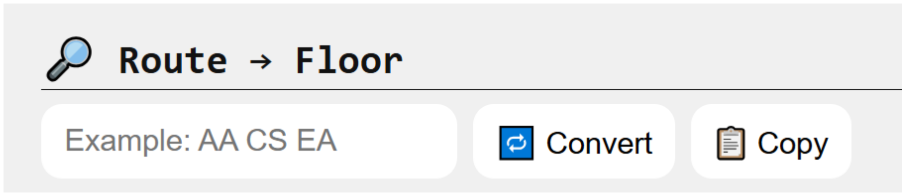
- **Header**

  🏢 **Facility** picker

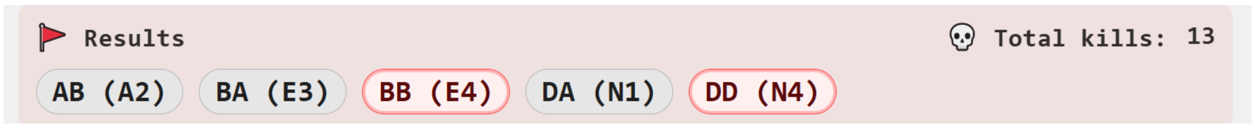
  Switching facilities **fully resets** the screen (results, chips, selections, counters).

- **Route** → **Floor** panel

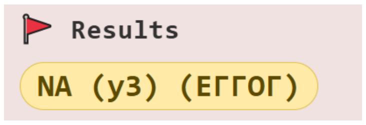
  

  - Input accepts **multiple routes** separated by **spaces** (or commas/semicolons/|).
  - **Copy** (📋) copies the current results line.

- 🚩 **Results** panel

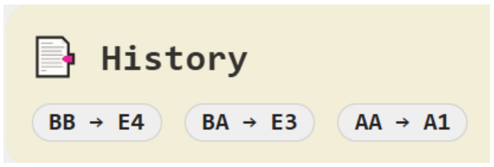
  - Shows chips for converted routes (e.g., `AA (A1)`).
  - **Warnings** (glove-typo fix) appear as emphasized chips, and the **red**


Results

NA (y3) (ЕГГОГ)

   - **ЕГГОГ** label marks a valid route whose **floor is not present** in the current layout (won't highlight on the map, won't count as a kill).

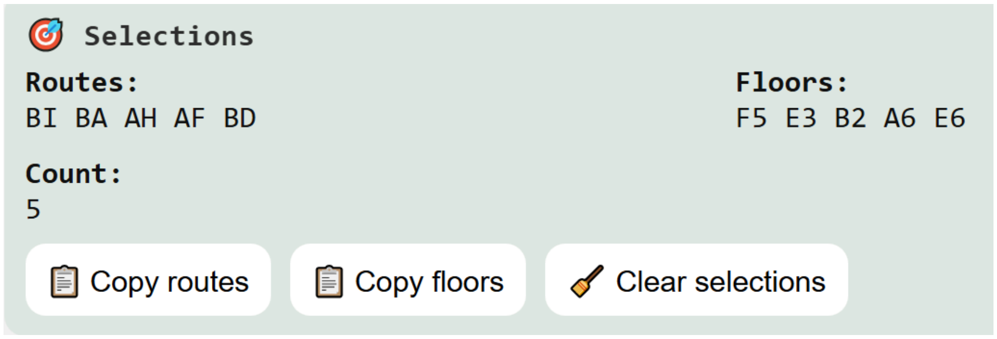
- 📑 **History** panel


History

BB → E4 | BA → E3 | AA → A1

   - Recent conversions as compact chips (latest first).

- 🎯 **Selections** panel


Selections

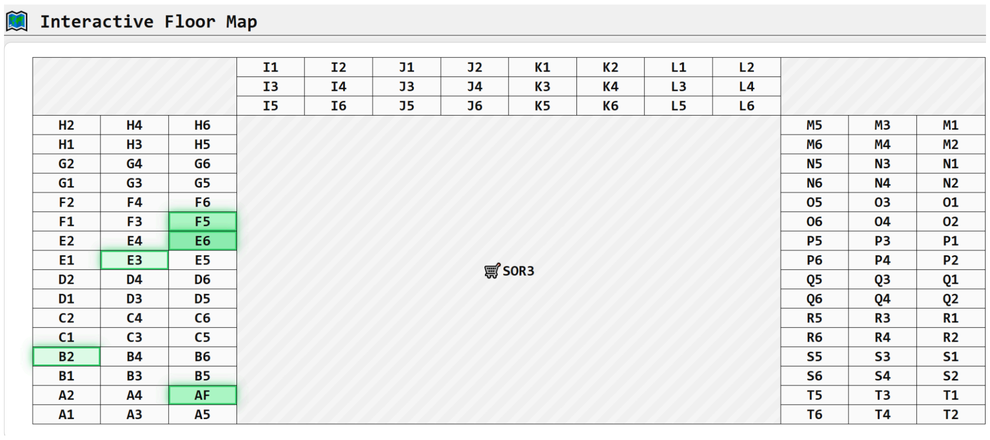
Routes:
BI BA AH AF BD

Floors:
F5 E3 B2 A6 E6

Count:
5

   - Click any map cell to **add/remove** it from a selection list.
   - Live counters and separate **Copy routes / Copy floors** buttons.

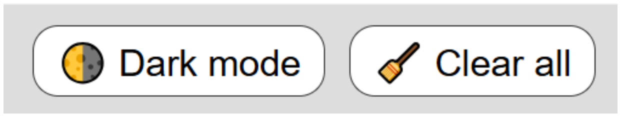
- 🗺️ **Interactive Floor Map**


Interactive Floor Map

			I1	I2	J1	J2	K1	K2	L1	L2			
			I3	I4	J3	J4	K3	K4	L3	L4			
			I5	I6	J5	J6	K5	K6	L5	L6			
H2	H4	H6									M5	M3	M1
H1	H3	H5									M6	M4	M2
G2	G4	G6									N5	N3	N1
G1	G3	G5									N6	N4	N2
F2	F4	F6									O5	O3	O1
F1	F3	F5									O6	O4	O2
E2	E4	E6									P5	P3	P1
E1	E3	E5				🛒SOR3					P6	P4	P2
D2	D4	D6									Q5	Q3	Q1
D1	D3	D5									Q6	Q4	Q2
C2	C4	C6									R5	R3	R1
C1	C3	C5									R6	R4	R2
B2	B4	B6									S5	S3	S1
B1	B3	B5									S6	S4	S2
A2	A4	AF									T5	T3	T1
A1	A3	A5									T6	T4	T2

   - Hover a cell: shows the **route**; otherwise shows the **floor** label.
   - Matching cells from conversion **blink** red/green (or both if also selected).

- **Footer**



- **Dark mode** toggle.
- **Clear all** button (same as **Esc**).

Daily Ops — How to Use in the Field

1. **Pick the facility** from the header.
2. **Enter routes** (e.g., `AA CS EA`) and press **Enter**.
3. **Watch the map**:
   - Matching cells **blink**; hovering a cell toggles route vs. floor text.
4. **Fix hints & warnings**:
   - Extra letters (e.g., `ADD`, `ADF`) are trimmed; you'll see a **red blinking** message: **`fixed ADD → AD, fixed ADF → AD`**.
   - Digits at the end are ignored with a small note: `digits ignored`.
   - If a floor doesn't exist on this layout, you'll get `(ЕГГОГ)` in Results.
5. **Selections**:
   - Click cells to build a selection set (green blink). Copy routes/floors via buttons.
6. **Reset**:
   - Press **Esc** or **Clear all**.

Keyboard & pointers

- **Enter**: convert current input
- **Esc**: full clear
- **Hover**: temporarily show route instead of floor
- **Click cell**: toggle selection

Route → Floor Conversion (How it works)

"Idiot-proof" input handling

- Input is normalized to **uppercase**.
- Any **digits at the end** of a token are **ignored** (and noted as `digits ignored` if applicable).
- Any **extra letters beyond the first two** (e.g., `AFF`, `ADF`) are **ignored**, and the token is flagged as a **glove-typo fix**:
    - Chip appears emphasized; and a red blinking note shows `fixed AFF → AF`.
- Duplicate routes in the same submission are **de-duplicated**.
- Tokens that **don't parse** as routes are labeled ``.

Layout presence check

- If a route converts to a floor that **doesn't exist in the current layout**, the tool returns a chip **\`ΕΓΓΟΓ\`** and **does not** highlight or count it.

Mathematical mapping for nerds (SOR3 example)

For SOR3, the logic file is `logic/SOR3.txt` containing the regex `([A-Z])([1-6])`. The second group `[1-6]` determines the number of **columns** `C=6` used by the mapping.

Let the two-letter route be αβ with `α,β ∈ {A,…,Z}`.

1. Convert letters to numbers:
   `a = ord(α) − ord(A)`, `b = ord(β) − ord(A)` with `a,b ∈ {0,…,25}`.

2. Linear index:
   `i = 26·a + b`.

3. With `C = 6` columns (SOR3):
   `row = ⌊ i / 6 ⌋`, `col = (i mod 6) + 1`.

4. Floor code:
   `Floor = chr(ord(A) + row)` // `col` (letter for row, 1-based number for column).

Checks: `AA → A1`, `AF → A6`, `AG → B1`.

For other facilities, `C` is inferred from the logic regex (e.g., `[1-3] → C=3`). The same formulas apply with your facility's `C`.

Adding a New Facility for even more nerds (end-to-end)

Quick Start

File layout (root of the tool):

```
/index.html
/list.txt                # facility codes, one per line (e.g., SOR3, PDX9)
/layouts/                # facility layouts as HTML tables
  ├── SOR3.txt
  └── PDX9.txt
/logic/                  # facility logic, 1 regex per file
  ├── SOR3.txt            # e.g., ([A-Z])([1-6])
  └── PDX9.txt            # e.g., ([A-Z])([1-3])
/logos/
  ├── SOR3.png
  └── PDX9.png
```

Serve locally (avoid `file://` fetch restrictions):

```
python -m http.server 8080
```

Create/Update `` (root):

- One facility code **per line**, uppercase recommended.
- Example:

  ```
  SOR3
  PDX9
  DPD1
  ```

- The dropdown order follows the file order. Lines starting with `#` are ignored (comments).

Create the logic file in `/logic/<FACILITY>.txt`:

- Content: a **single regex** that defines valid floor labels, e.g.:
  - `([A-Z])([1-6])` → floors like `A1…A6, B1…B6, …` (→ `C=6`)
  - `([A-Z])([1-3])` → floors like `A1…A3, B1…B3, …` (→ `C=3`)
- **Tips:**
  - Provide only the **pattern**, no surrounding slashes. The tool anchors it to `^…$` automatically.
  - The **second capture** must be the **column number** range; the tool infers column count `C` from ranges like `[1-6]`.

- Keep it simple; if your layout uses more complex numbering, align the regex accordingly.

Create the layout in `/layouts/<FACILITY>.txt`:

A **clean HTML** `` (no extraneous wrappers/styles). Example minimal grid:

```
<table>
  <tbody>
    <tr>

<td>A1</td><td>A2</td><td>A3</td><td>A4</td><td>A5</td><td>A6</td
>
    </tr>
    <tr>

<td>B1</td><td>B2</td><td>B3</td><td>B4</td><td>B5</td><td>B6</td
>
    </tr>
    <!-- ...more rows as needed... -->
  </tbody>
</table>
```

You may include `rowspan/colspan` cells (e.g., staging) or **blank** cells. To mark a decorative/empty area, leave the `<td>` empty; the tool styles it as a blank.

Do not wrap labels inside `<p>`, `<span>`, etc. Just `A1`, `B3`, … inside the `<td>`.

1. **Add a logo** to `/logos/<FACILITY>.png`:

   Small horizontal image works best (it sits in the header).
2. **Test**:

   Start local server; open the tool; pick the new facility.
   Enter test routes (e.g., `AA AF AG`).
   Confirm that highlights appear in expected cells. If not:

   - Check that table cells exactly match the **regex format** (e.g., `A1`, not `a1` or `A01`).
   - Verify the **column range** in the logic file matches the **number of columns** you intend to map (e.g., `[1-3]` → 3 columns).

Governance & Conventions

- **File naming**: Use **exact facility code** (e.g., `SOR3.txt`, `SOR3.png`).

- **Regex logic**: Keep the second capture group as the **numeric column**; prefer simple ranges like `[1-6]`.
- **Layout table**: Keep cells **plain** (`A1`, `B2`, `…`). Avoid wrapping tags or inline styles.
- **Localization**: UI text and comments are in **English**.
- **Reset on switch**: Facility switching performs a full state reset by design.

Appendix — Developer Notes

- **Logic ingestion**
  - The tool reads `/logic/<FACILITY>.txt` and normalizes it to a full-string regex (`^…$`). Column count `C` is inferred from a range like `[1-6]` (fallback `C=6`).
- **Mapping math (generalized)**
  - With `C` columns from logic, for route αβ:
    - `i = 26·(ord(α)−ord(A)) + (ord(β)−ord(A))`
    - `row = ⌊ i / C ⌋`, `col = (i mod C) + 1`
    - `Floor = chr(ord(A)+row) ∥ col`
- **Layout parsing**
  - Cells matching the logic regex become interactive (`data-floor`, `data-route`). Empty non-stage cells get a “blank” style.
- **Warnings**
  - Extra letters beyond two are flagged; duplicates are deduped; invalid tokens are labeled `(invalid)`.
- **Missing-in-layout**
  - Valid `route→floor` that lacks a matching cell renders `(EГГOГ)` and is excluded from kill counters.

Contact

- Owner: **Rio (@rifenris)** — 10@r10.wtf