Hubitat Tile Mover

A command-line utility to make mass changes to tile layout in Hubitat Dashboard layout JSON:

Overview:

Import, modify and output Hubitat dashboard layouts:

- **Import** dashboard layouts directly from the hub • JSON files • the clipboard (default)

- **Output** changed layouts directly back to the hub • JSON files • the clipboard (default)

Layout Actions

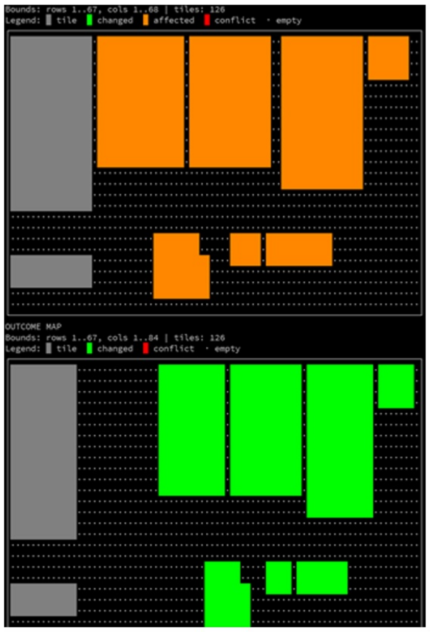
- **MOVE** tiles: columns, rows, range
- **COPY** tiles: columns, rows, range
- **MERGE** tiles (copy from another dashboard): columns, rows, range
- **INSERT** full or partial columns, rows (push tiles over/down at column/row)
- **DELETE** full or partial columns, rows (remove tiles and pull tiles left or up)
- **CLEAR** tiles (remove but keep layout): columns rows, range
- **CROP** layout (remove all tiles not in): columns, rows, range
- **PRUNE** layout (remove all tiles listed or all tiles ***except*** those listed): tile id's, device ids
- **TRIM** layout (remove blank rows, cols): top, left

Features

- Preserve, duplicate or remove custom CSS rules when tiles added (copied) or removed by actions

- Prevent actions which would result in tiles be placed over existing tiles

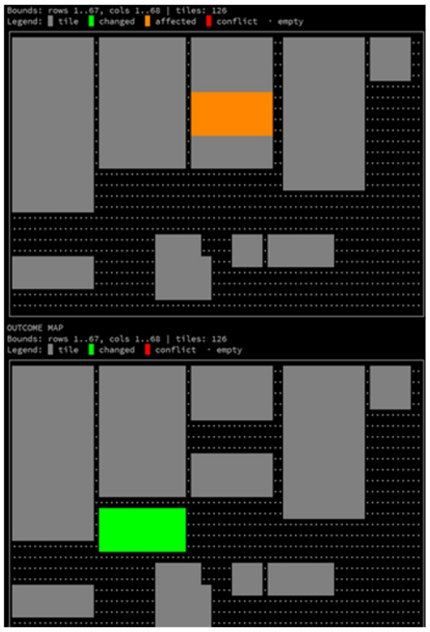
- Visual Layout Maps: Easily see proposed changes, potential conflicts and final outcome of actions



```
Bounds: rows 1..67, cols 1..68 | tiles: 126
Legend: ▮ tile ▮ changed ▮ affected ▮ conflict · empty

OUTCOME MAP
Bounds: rows 1..67, cols 1..84 | tiles: 126
Legend: ▮ tile ▮ changed ▮ conflict · empty
```

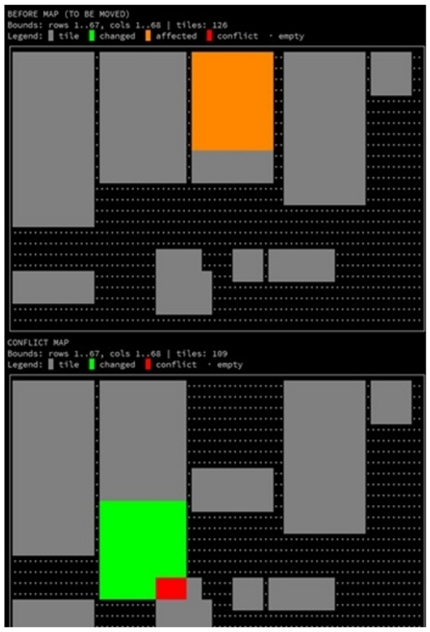
Insert Columns



```
Bounds: rows 1..67, cols 1..68 | tiles: 126
Legend: ▮ tile ▮ changed ▮ affected ▮ conflict · empty

OUTCOME MAP
Bounds: rows 1..67, cols 1..68 | tiles: 126
Legend: ▮ tile ▮ changed ▮ conflict · empty
```

Move Range



```
BEFORE MAP (TO BE MOVED)
Bounds: rows 1..67, cols 1..68 | tiles: 126
Legend: ▮ tile ▮ changed ▮ affected ▮ conflict · empty

CONFLICT MAP
Bounds: rows 1..67, cols 1..68 | tiles: 109
Legend: ▮ tile ▮ changed ▮ conflict · empty
```

Layout Conflict Detection

Layout Import Sources and Output Destinations:

Import Sources

Exactly one import method is used. If not specified, clipboard is the default.

- Valid Sources:
  - `--import:clipboard` — Read JSON text from clipboard.
  - `--import:file <filename>` — Read JSON text from file.
  - `--import:hub` — Fetch the full layout JSON from Hubitat using `--url`.

- Required for `--import:hub`:
  - `--url "<dashboard_local_url>"`

- Dashboard URL format (typical):

```
http://<hub-ip>/apps/api/<appId>/dashboard/<dashId>?access_token=
<token>&local=true
```

Output destinations

- Valid Destinations:
  - `--output:terminal` — Print output to terminal.
  - `--output:clipboard` — Write to clipboard (default)
  - `--output:file <filename>` — Write to file.
  - `--output:hub` — POST full layout JSON back to Hubitat using `--url`.

- Required for `--output:hub`:
  - `--url "<dashboard_local_url>"`

- Notes:
  - Output will be set to the clipboard if `--output:<destination>` is not specified or present.
  - `--output:hub` will fail if:

        - Import does not contain the full layout JSON object.
        - `--url` is not a valid local dashboard url or cannot be reached.
        - A valid requestToken could not be obtained.

Output format (level)

Allows down-level output (full → minimal → bare). If omitted, output defaults to match input.

- `--output_format:full`
- `--output_format:minimal`
- `--output_format:bare`

Layout Actions Overview:

Action types:

1. **Primary edit operation** (at most one per run):
   insert, move, copy, merge, delete, clear, crop, prune
2. Supplemental actions (can run alone or after a primary operation):

- maps: `--show_map`
- trim: `--trim` / `--trim:top` / `--trim:left`
- sort: `--sort:<spec>`
- scrub CSS: `--scrub_css`

`--undo_last` is standalone and supersedes all other actions.

Action targets:

- Tile location is determined upper left corner row and column.
- Most actions are applied to only to dashboard tiles located within the target columns, rows or range.
- Tile span is the space a tile occupies calculated as row + height -1, column + width -1.
- Tiles located outside of the target but whose span extends into it, are considered to be overlapping the target, and are not selected.
- Use the `--include_overlap` switch to include tiles that overlap the target columns, rows or range.

Layout Actions

Insert

Inserts empty whole or partial rows or columns by pushing tiles beyond the insertion point

- Actions:
  - `--insert_rows <count> <at_row>` — Pushes down (increase tile's 'row' by `<count>`) tiles at/after `<at_row>`, and optionally tiles overlapping the insertion row.
  - `--insert_cols <count> <at_col>` — Pushes right (increase tile's 'col' by `<count>`) tiles at/after `<at_col>`, and optionally tiles overlapping the insertion column.
- Selection Modifiers:
  - `--col_range <start_col> <end_col>` — Insert rows only in column range. Only valid with `--insert_rows`
  - `--row_range <start_row> <end row>` — Insert columns only in row range. Only valid with `--insert_cols`
  - `--include_overlap`
- Options:
  - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

Move

Moves tiles to a new location.

- Actions:
  - `--move_cols <start_col> <end_col> <dest_start_col>`
  - `--move_rows <start_row> <end_row> <dest_start_row>`

    - `--move_range <src_top_row> <src_left_col> <src_bottom_row> <src_right_col> <dest_top_row> <dest_left_col>`

- Selection Modifiers:
    - `--include_overlap`

- Options:
    - `--allow_overlap`
      ignore conflicts and allow moved tiles to overlap existing tiles at the destination.
    - `--skip_overlap`
      skip tiles that would conflict, move all others.

    - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

- Notes:
    - Conflict detection is evaluated **once, *before*** moving/copying, against **existing destination tiles only**. Any tiles that are being copied/moved can be overlapped and will not be considered in conflict.
    - Default behavior: if any conflicts exist, abort before moving anything.

Copy

Same as Move, but originals remain. Copies are created with new IDs. Existing tile specific CSS rules in customCSS can be optionally copied with the new IDs

- Actions:
    - `--copy_cols <start_col> <end_col> <dest_start_col>`
    - `--copy_rows <start_row> <end_row> <dest_start_row>`
    - `--copy_range <src_top_row> <src_left_col> <src_bottom_row> <src_right_col> <dest_top_row> <dest_left_col>`

- Selection Modifiers:

    - `--include_overlap`

- Options:
    - `--allow_overlap` — ignore conflicts and allow moved tiles to overlap existing tiles at the destination.
    - `--skip_overlap` — skip tiles that would conflict, move all others.

    - `--ignore_css` — disables creating/copying CSS for new IDs.

    - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

- Notes:
    - ID allocation for new tiles: New IDs are created sequentially beginning starting with 1 + max(highest existing tile ID, highest referenced tile ID in customCSS). This prevents any orphaned CSS rules from being applied to new tiles.
    - By default, customCSS is checked for any tile specific CSS rules for copied tiles. Any rules found are duplicated for the new tile id
    - Conflict detection is evaluated **once, *before*** moving/copying, against **existing destination tiles only**.
    - Tiles that are being copied/moved can be overlapped and will not be considered in conflict.
    - Actions will be aborted if conflicts are found unless `--allow_overlap` or `--skip_overlap` is present.

Merge

Copy tiles from another dashboard layout into this layout.

- Actions:
    - `--merge_cols <start_col> <end_col> <dest_start_col>`
    - `--merge_rows <start_row> <end_row> <dest_start_row>`
    - `--merge_range <src_top_row> <src_left_col> <src_bottom_row> <src_right_col> <dest_top_row> <dest_left_col>`

- Required: source selection:

    - `--merge_source <filename>` — load source JSON from file
    - `--merge_url "<other_dashboard_local_url>"` — fetch source JSON from hub

- Selection Modifiers:

    - `--include_overlap`

- Options:

    - `--allow_overlap` — ignore conflicts and allow moved tiles to overlap existing tiles at the destination.

    - `--skip_overlap` — skip tiles that would conflict, move all others.

    - `--ignore_css` — disables creating/copying CSS for new IDs.

    - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

- Notes:

    - ID allocation for new tiles: New IDs are created sequentially beginning starting with 1 + max(highest existing tile ID, highest referenced tile ID in customCSS). This prevents any orphaned CSS rules from being applied to new tiles.
    - By default, customCSS is checked for any tile specific CSS rules for copied tiles. Any rules found are duplicated for the new tile id
    - Conflict detection is evaluated **once, *before*** moving/copying, against **existing destination tiles only**.
    - Tiles that are being copied/moved can be overlapped and will not be considered in conflict.
    - Actions will be aborted if conflicts are found unless `--allow_overlap` or `--skip_overlap` is present.

Delete

Deletes tiles located in the target rows or columns, then shifts remaining tiles to close the gap.

- Actions:

    - `--delete_rows <start_row> <end_row>`

- `--delete_cols <start_col> <end_col>`

- Selection Modifiers:
  - `--col_range <start_col> <end_col>` — Deletes rows only in column range. Only valid with `--insert_rows`
  - `--row_range <start_row> <end row>` — Deletes columns only in row range. Only valid with `--insert_cols`

  - `--include_overlap`

- Options:
  - `--force` — skip confirmation prompts -- assume yes.
  - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.
  - `--cleanup_css` — remove tile-specific CSS rules from 'customCSS' for tiles deleted by the current action.

- Notes:
  - Default behavior is to leave tile specific CSS rules for deleted tiles in place unless `--cleanup_css` is present.
  - Use the `--scrub_css` action to remove orphaned CSS rules for other tiles,

---

Clear

Removes tiles in the target rows, columns or range but does change the dashboard layout.

- Actions:
  - `--clear_rows <start_row> <end_row>`
  - `--clear_cols <start_col> <end_col>`
  - `--clear_range <top_row> <left_col> <bottom_row> <right_col>`
- Selection Modifiers:
  - `--include_overlap`

- Options:
  - `--force` — skip confirmation prompts -- assume yes.
  - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

- `--cleanup_css` — remove tile-specific CSS rules from 'customCSS' for tiles deleted by the current action.

- Notes:
    - Default behavior is to leave tile specific CSS rules for cleared tiles in place unless `--cleanup_css` is present.
    - Use the `--scrub_css` action to remove orphaned CSS rules for other tiles,

---

Crop

Clears tiles outside of the target rows, columns or range. The position of remaining tiles is unchanged.

- Actions:
    - `--crop_to_rows <start_row> <end_row>`
    - `--crop_to_cols <start_col> <end_col>`
    - `--crop_to_range <top_row> <left_col> <bottom_row> <right_col>`

- Selection Modifiers:
    - `--include_overlap`

- Options:
    - `--force` — skip confirmation prompts -- assume yes.
    - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.
    - `--cleanup_css` — remove tile-specific CSS rules from 'customCSS' for tiles deleted by the current action.

- Notes:
    - Default behavior is to leave tile specific CSS rules for cleared tiles in place unless `--cleanup_css` is present.
    - Use the `--scrub_css` action to remove orphaned CSS rules for other tiles,
    - At least one tile must exist in the target rows, columns or range as at least one tile must remain after cropping.

- Use `--trim`, `--trim:top` or `--trim:left` to remove blank rows on the top or columns on the left of the remaining tiles.

Prune

Clears all tiles listed, or all tiles ***except*** those listed in a comma separated list of tile ids or device ids. The position of remaining tiles is unchanged.

- Actions:
  - `--prune_ids "<id_list>"`
  - `--prune_devices "<device_list>"`

  - `--prune_except_ids "<id_list>"`
  - `--prune_except_devices "<device_list>"`

- Acceptable List Values:
  - Explicit values: `1,4,6,8,9`
  - Comparisons: `<29`
  - Inclusive ranges: `3-20,40-58`
  - Combination: `<29,43,46,>=100`

- Options:
  - `--force` — skip confirmation prompts -- assume yes.
  - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.
  - `--cleanup_css` — remove tile-specific CSS rules from 'customCSS' for tiles deleted by the current action.

- Notes:
  - Default behavior is to leave tile specific CSS rules for cleared tiles in place unless `--cleanup_css` is present.
  - Use the `--scrub_css` action to remove orphaned CSS rules for other tiles,
  - At least one matching tile id or device id must exist. At least one tile must remain after pruning.
  - Use `--trim`, `--trim:top` or `--trim:left` to remove blank rows on the top or columns on the left of the remaining tiles.

Trim

Removes blank rows above the top-most tile and/or blank columns left of the left-most tile.

- Actions:
  - `--trim` (defaults to top+left)
  - `--trim:top`
  - `--trim:left`
- Options:
  - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.
- Note:
  - The trim action can be used in conjunction with another action or as a standalone action..
  - When combined with another action, trimming will only occur after successful completion of the primary action.

Supplemental Actions and Options

Sort

Changes the order tiles appear in the dashboard layout JSON only

- Actions:
  - `--sort:<spec>`
- Sort Keys:
  - `[-]i` = id
  - `[-]r` = row
  - `[-]c` = col
  - Sort order for keys is ascending unless preceded by a "`-`"
- Notes:
  - Sorting only changes the order tiles are listed in the layout JSON. It has no effect on the order tiles appear on the dashboard.
  - By default, actions do not change the order tiles are listed in the layout JSON unless `--sort` is present.
  - The default sort order is `irc`(id, row, column) in ascending order.

    - Ascending or descending order can be specified for each key. Examples: `--sort:i-rc` or `--sort:-ir-c`.
    - Missing sort keys are assumed. For example `--sort:r` will result in `--sort:ric`.
    - No sorting is applied to CSS rules in customCSS.

Visual Layout Maps

Show before, outcome and conflict layout previews in the terminal

- Optional Action:

    - `--show_map` — Enables generation of dashboard layout maps

- Map Options:

    - `--map_focus:full` — All maps are zoomed out to show the full dashboard, scaled to fit the terminal.
    - `--map_focus:no_scale` — All maps are zoomed out to show an unscaled view of the full dashboard. Each row / column is represented by one character space and may not display properly depending on terminal size.
    - `--map_focus:conflict` — Conflict maps are zoomed in to show just the tiles in conflict. All other maps are zoomed out to show the full dashboard. All maps are scaled.

- Map Legend:

  · (gray dot) - empty spaces
  ⬜ (gray) - unaffected tiles
  🟧 (orange) - tiles in the target row, column or range before changes are made
  🟩 (green) - tiles successfully changed by the action or portions not in conflict.
  🟥 (red) - tiles (or portions) in conflict that caused the action to fail.
  🟨 (yellow) - tiles (or portions) conflicts allowed by `--allow_overlap`

Miscellaneous Options:

- Prompts / Confirmation Options:

    - `--force` — Suppress confirmation prompts for actions which remove tiles or custom CSS rules.

    - `--confirm_keep` — After writing output, prompts (independently of `--force`) to keep or undo the changes made.

- Safety / Undo Options:

    - `--lock_backup` — Retains the last undo backup (if found) as the undo backup for the current action.
    - `--undo_last` — Load the undo backup (if found) and writes it to the previous actions output destination.

- Output / Debug Information Options:
    - `--quiet` — suppress end-of-run summary line (errors still shown)
    - `--verbose` — planned actions + concise results
    - `--debug` — per-tile action logs + deep details

- Notes:
    - The undo backup contains the JSON imported for an action. It is only created after an action has completed and was successfully saved to the output destination.
    - The purpose of `--confirm_keep` is to provide an opportunity to review or test the outcome of an action, then if necessary, undo it. This is useful when tweaking action target ranges or options. It is the same as running an action without `--confirm_keep`, then using the `--undo_last` action.

Help

- `-h`, `--help` — Short help
- `--help_full` — Full detailed help

Examples:

- Insert 2 columns at col 15 (only in rows 4–32):

  ```
  python hubitat_tile_mover.py --insert_cols 2 15 --row_range 4 32
  ```

- Move columns 1–14 to start at 85 and save back to hub. Show the layout before and after maps:

  ```
  python hubitat_tile_mover.py --import:hub --url "<dashboard_local_url>" --
  move_cols 1 14 85 --output:hub --show_map
  ```

- Copy tiles in the rectangular range 1,1 to 2,20 to a new location at 40,40:

  ```
  python hubitat_tile_mover.py --copy_range 1 1 20 20 40 40
  ```

- Crop to a range (delete everything outside), show maps, force, cleanup CSS, save to a file:

  ```
  python hubitat_tile_mover.py --crop_to_range 1 1 85 85 --show_map --force --
  cleanup_css --output:file "<filename.json>"
  ```

- Remove orphan CSS rules only (no tile edits) without confirmation prompt:

  ```
  python hubitat_tile_mover.py --scrub_css --force
  ```

- Undo last run (restore last input to last outputs unless overridden):

  ```
  python hubitat_tile_mover.py --undo_last
  ```

Batch Actions:

Tips for running a batched or consecutive actions:

- Use `--lock_backup` on all actions except the first action. This provides an undo from before any actions in the batch were run.
- Reduce risk and increase speed by minimizing using the clipboard to read and write intermediate layouts in batches.

Example Batch

Batch Overview

- Get the dashboard layout from the hub and insert 5 empty columns at column 10
- Merge (copy) columns 15 - 20 from another dashboard into columns 10 - 15 of this dashboard.
- Crop the resulting layout keep only tiles in the rectangular region 10,10 - 40,30
- Trim empty rows left behind at the top and left sides
- Cleanup orphaned CSS rules and output the final layout back to the hub.

Example

```
python hubitat_tile_mover.py --import:hub --url "<dashboard_local_url>" --
output:clipboard --insert_cols 5 10
python hubitat_tile_mover.py --import:clipboard --output:clipboard --merge_url: "
<other_dashboard_url>" --merge_cols 15 20 10 --lock_backup
python hubitat_tile_mover.py --import:clipboard --url "<dashboard_local_url>" --
output:hub --crop_to_range 10 10 40 30 --include_overlaps --force --cleanup_css --
trim --lock_backup
```

Batched Actions in Detail

First Action

1. Imports a layout from the hub

2. Inserts 5 blank columns at column 10 in the layout

3. Writes the new layout to the ***clipboard***

4. Saves the original imported layout to the undo backup file. In the event of an error, running `undo_last` would not be necessary to undo any actions performed by the batch as they have only been saved to the

clipboard. The original layout still exists unchanged on the hub.

The second action:

1. Imports the layout saved by the first action from the clipboard.
2. Copies tiles in columns 15-20 from another dashboard into the blank columns created by the previous action at column 10
3. Writes the new layout to the ***clipboard***
4. Does not change the undo backup created by the first action. In the event of an error, running `undo_last` would not be necessary to undo any actions performed by the batch as they have only been saved to the clipboard. The original layout still exists unchanged on the hub.

The third action:

1. Imports the layout saved by the second action from the clipboard.
2. Removes all dashboard tiles except those located or having some portion inside the rectangular range of (row 10, col 10) to (row 40, col 30).
3. Does present a confirmation prompt.
4. Removes any tile specific CSS rules from "customCSS" for the tiles that were removed.
5. Again, does not present a confirmation prompt.
6. Trims blank column and rows from the top and left sides to move the layout of the remaining tiles as far to the upper left as possible.
7. Writes the final new layout to the ***hub***
8. Does not change the undo backup created by the first action. In the event of an error, running `--undo_last` would undo all changes made by the batch by restoring the undo backup made in the first action.
