# Tile Builder Help
# Version 1.0

# Table of Contents

# Section 1

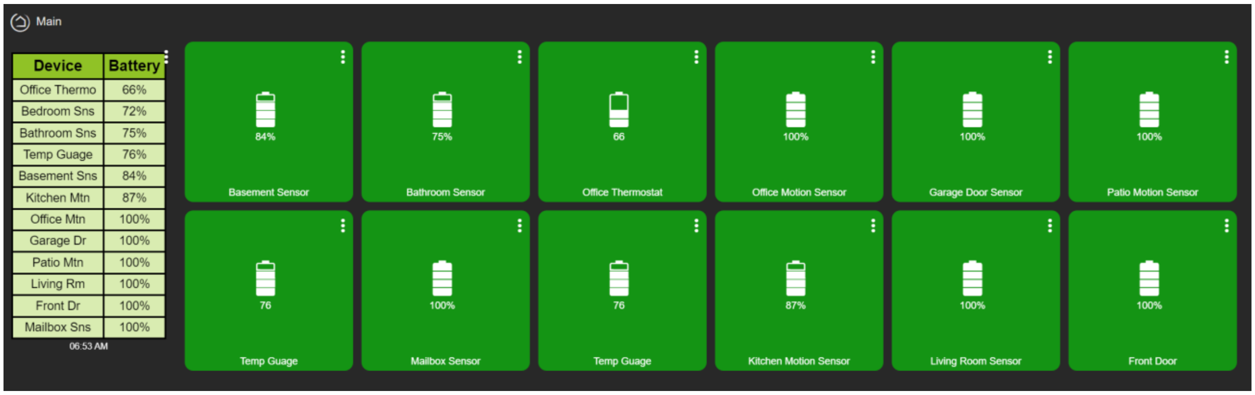
# Tile Builder

# Introduction and Installation

## Introduction

Tile Builder is a novel way of presenting data on a Hubitat Dashboard.  Rather than each tile being a single unique device, **Tile Builder** allows data from multiple devices to be presented on the same tile in a highly customizable tabular format.

It is entirely native to Hubitat and does not require any third-party elements or knowledge of CSS to achieve quite impressive results. The image below shows one of these tables vs the traditional method of using individual tiles.



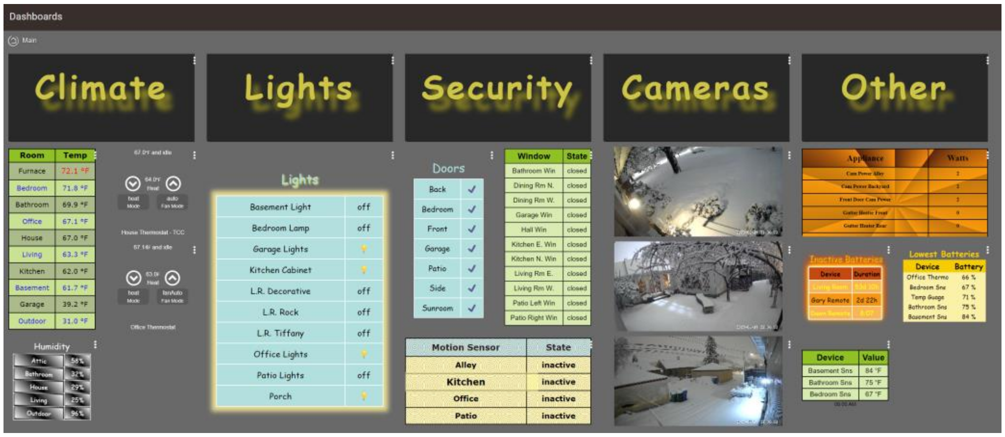
| Device | Battery |
|---|---|
| Office Thermo | 66% |
| Bedroom Sns | 72% |
| Bathroom Sns | 75% |
| Temp Guage | 76% |
| Basement Sns | 84% |
| Kitchen Mtn | 87% |
| Office Mtn | 100% |
| Garage Dr | 100% |
| Patio Mtn | 100% |
| Living Rm | 100% |
| Front Dr | 100% |
| Mailbox Sns | 100% |

06:53 AM

### Main Benefits of Tile Builder

- A native solution for Hubitat dashboards that is easy to use and looks great.
- Full control over color, style, placement, effects of tiles etc.
- Data can be presented at a much higher density. Great for tablets and phones.
- Does not require maintenance of a separate external platform.
- Entirely local to the hub. No need to send events off to a remote system.

Devices that only present data without a control option are prime candidates. Good examples are temperature, motion, pressure, humidity, battery, contacts, presence, leaks etc. The example below is built entirely using **Tile Builder Advanced** except for the 2 thermostats and the 3 camera image. I place all the action items on separate dashboards which are only used when something does not seem right on the main page. The assorted styles show the level of control available in a **Tile Builder** tile.



| Room | Temp |
|---|---|
| Furnace | 72.1 °F |
| Bedroom | 71.8 °F |
| Bathroom | 69.9 °F |
| Office | 67.1 °F |
| House | 67.0 °F |
| Living | 63.3 °F |
| Kitchen | 62.0 °F |
| Basement | 61.7 °F |
| Garage | 39.2 °F |
| Outdoor | 31.0 °F |

Humidity

| | |
|---|---|
| Attic | 56% |
| Bathroom | 32% |
| House | 29% |
| Living | 25% |
| Outdoor | 96% |

Lights

| | |
|---|---|
| Basement Light | off |
| Bedroom Lamp | off |
| Garage Lights | 💡 |
| Kitchen Cabinet | 💡 |
| L.R. Decorative | off |
| L.R. Rock | off |
| L.R. Tiffany | off |
| Office Lights | 💡 |
| Patio Lights | off |
| Porch | 💡 |

Doors

| | |
|---|---|
| Back | ✓ |
| Bedroom | ✓ |
| Front | ✓ |
| Garage | ✓ |
| Patio | ✓ |
| Side | ✓ |
| Sunroom | ✓ |

| Window | State |
|---|---|
| Bathroom Win | closed |
| Dining Rm N. | closed |
| Dining Rm W. | closed |
| Garage Win | closed |
| Hall Win | closed |
| Kitchen E. Win | closed |
| Kitchen N. Win | closed |
| Living Rm E. | closed |
| Living Rm W. | closed |
| Patio Left Win | closed |
| Patio Right Win | closed |

| Motion Sensor | State |
|---|---|
| Alley | inactive |
| Kitchen | inactive |
| Office | inactive |
| Patio | inactive |

| Appliance | Watts |
|---|---|
| Cam Power Alley | 2 |
| Cam Power Backyard | 2 |
| Front Door Cam Power | 2 |
| Gutter Heater Front | 0 |
| Gutter Heater Rear | 0 |

Inactive Batteries

| Device | Duration |
|---|---|
| [illegible] | [illegible] |
| Gary Remote | 2d 22h |
| [illegible] | [illegible] |

Lowest Batteries

| Device | Battery |
|---|---|
| Office Thermo | 66 % |
| Bedroom Sns | 67 % |
| Temp Guage | 71 % |
| Bathroom Sns | 75 % |
| Basement Sns | 84 % |

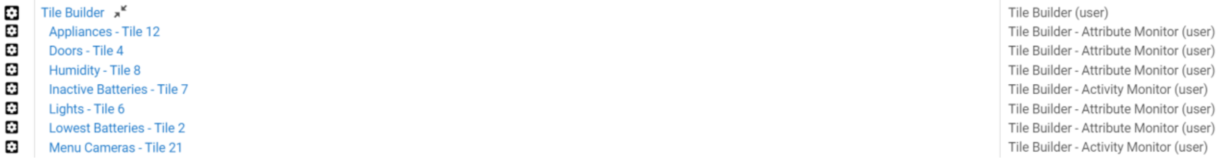
| Device | Value |
|---|---|
| Basement Sns | 84 °F |
| Bathroom Sns | 75 °F |
| Bedroom Sns | 67 °F |

## How Tile Builder Works

There are four components to **Tile Builder,** all of which are required for operation.

1) **Tile Builder Parent App**
2) **Attribute Monitor** (child app) – Generates device tables using a single attribute such as the first battery example.
3) **Activity Monitor** (child app) – Generates device tables using the 'Last Activity' attribute and can be used to monitor both active and inactive groups of devices.
4) **Tile Builder Storage Driver** – Device driver used for storing **Tile Builder** data.

The Tile Builder parent app is the primary organizing app.



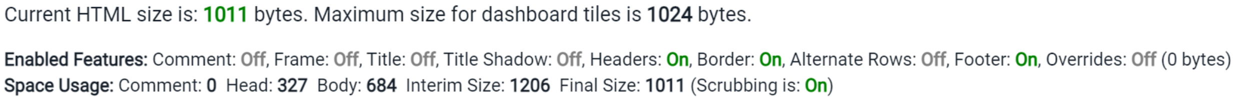
| | |
|---|---|
| Tile Builder | Tile Builder (user) |
| Appliances - Tile 12 | Tile Builder - Attribute Monitor (user) |
| Doors - Tile 4 | Tile Builder - Attribute Monitor (user) |
| Humidity - Tile 8 | Tile Builder - Attribute Monitor (user) |
| Inactive Batteries - Tile 7 | Tile Builder - Activity Monitor (user) |
| Lights - Tile 6 | Tile Builder - Attribute Monitor (user) |
| Lowest Batteries - Tile 2 | Tile Builder - Attribute Monitor (user) |
| Menu Cameras - Tile 21 | Tile Builder - Activity Monitor (user) |

Tiles are generated by one of the two child apps and organized under the parent app. When tiles are generated, the results are stored in the **Tile Builder Storage Device** in a named tile attribute (tile1 – tile25). This attribute is then placed onto the dashboard as shown previously.

**Attribute Monitor** uses an event subscription model. If the value of one of the monitored attributes changes the table is automatically regenerated and is updated on the dashboard. This happens within fractions of a second without any perceptible delay.

**Activity Monitor** only relates to the **Last Activity** time on a device and uses a timed model. A query is constructed via the child app such as **Inactive Battery Devices** to list only those battery devices that have had no activity in over 24 hours for example. A refresh interval is assigned from minutes to hours to months. When that interval is reached the table is regenerated, the results are stored in the **Tile Builder Storage Device** and immediately displayed in the dashboard.

The Hubitat dashboard has a limit of 1,024 bytes for any attributes that are display and **Tile Builder** operates within this constraint without tricks or workarounds. The simple rule of thumb is the more data you want to display the less features can be utilized. But don't worry, **Tile Builder** offers a great deal of optimization and guidance to help you find that balance point. When you build a tile the size of the tile and which components are active is always displayed along the bottom.



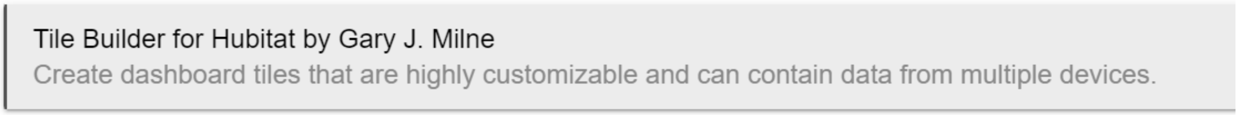
Current HTML size is: **1011** bytes. Maximum size for dashboard tiles is **1024** bytes.

**Enabled Features:** Comment: Off, Frame: Off, Title: Off, Title Shadow: Off, Headers: **On**, Border: **On**, Alternate Rows: Off, Footer: **On**, Overrides: Off (0 bytes)
**Space Usage:** Comment: **0** Head: **327** Body: **684** Interim Size: **1206** Final Size: **1011** (Scrubbing is: **On**)

I have been able to display up to 20 devices in the most basic of tables. But if you want a moderate level of customization then 8-12 devices is probably more realistic. With all display features turned on, including highlights and animations (Advanced Version), 4-8 devices is probably a reasonable estimate.

## Tile Builder Installation

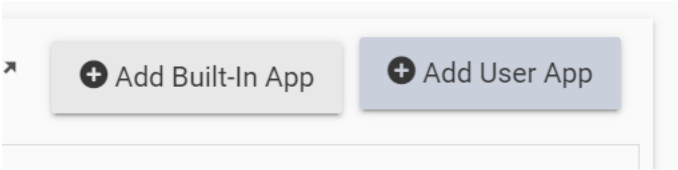
**Tile Builder** is listed in Hubitat Package manager. Choose to install by tags and select the **Dashboards** tag. Select **Tile Builder for Hubitat** and complete the installation process. This will place the code on your hub and then there are a few steps to complete the installation.



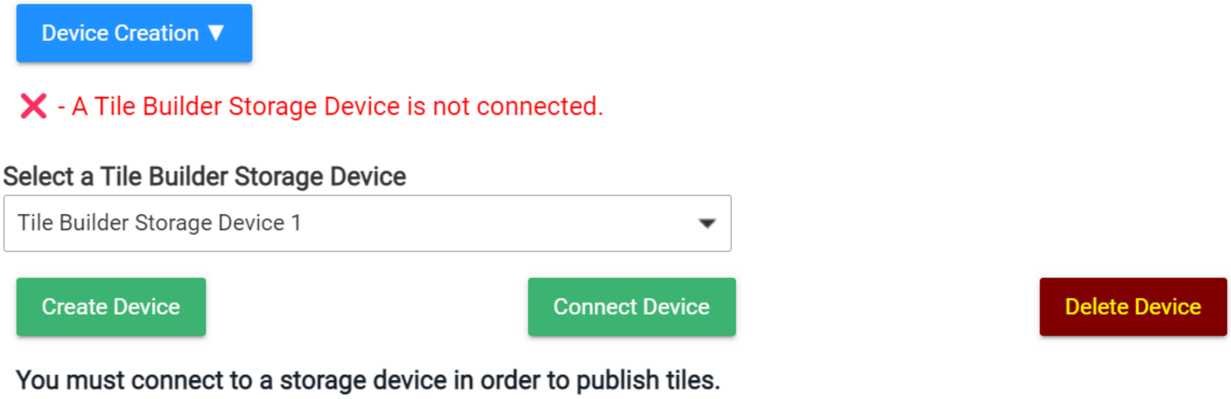
Tile Builder for Hubitat by Gary J. Milne

Create dashboard tiles that are highly customizable and can contain data from multiple devices.

1. Go to the Apps tab and click on **Add User App**



2. Select **Tile Builder** from the list of available apps.
3. **Tile Builder** will install and bring you to the parent screen.
4. **Select the appropriate License Type (see paragraph below on licensing).**
5. Under **Device Creation** you must first create the storage device and then connect the app to the device. Just use the default device for now.



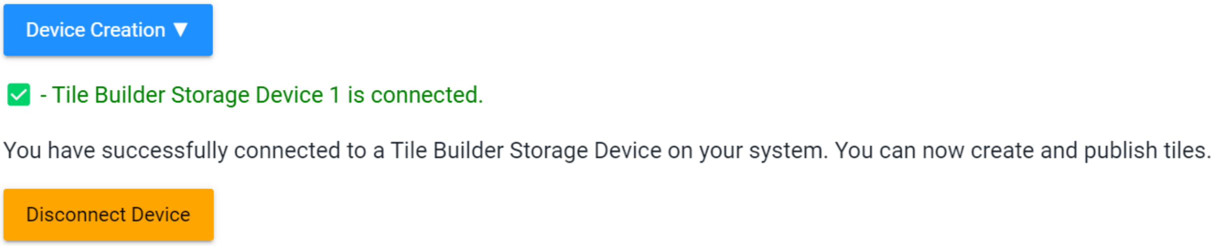
❌ - A Tile Builder Storage Device is not connected.

Select a Tile Builder Storage Device

Tile Builder Storage Device 1

You must connect to a storage device in order to publish tiles.

6. Once the device is created and connected it will look like this.



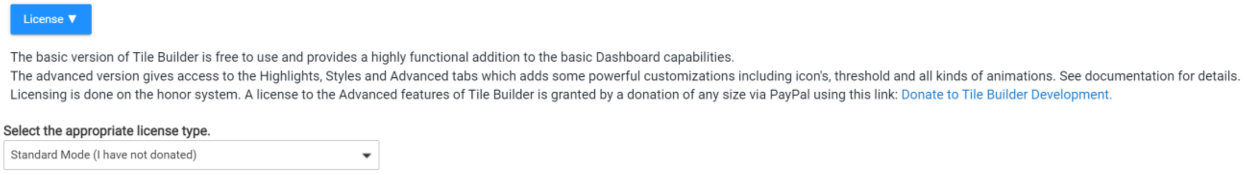
☑ - Tile Builder Storage Device 1 is connected.

You have successfully connected to a Tile Builder Storage Device on your system. You can now create and publish tiles.

7. We can now create our first tile.

## Licensing

**Tile Builder** has Standard and Advanced versions. The **Standard** version has a great degree of functionality and is entirely free to use as much as you wish. The **Advanced** version adds powerful features such as highlighting, thresholds, styles, overrides and a whole lot more.



The basic version of Tile Builder is free to use and provides a highly functional addition to the basic Dashboard capabilities.
The advanced version gives access to the Highlights, Styles and Advanced tabs which adds some powerful customizations including icon's, threshold and all kinds of animations. See documentation for details.
Licensing is done on the honor system. A license to the Advanced features of Tile Builder is granted by a donation of any size via PayPal using this link: Donate to Tile Builder Development.

Select the appropriate license type.
Standard Mode (I have not donated)

A lifetime license for the Advanced version is currently $5 and is on the honor system. When you make a payment via the link DM me on the Hubitat community and I will add you to my list of registered users.

The top 5 reasons to be a registered user are as follows:

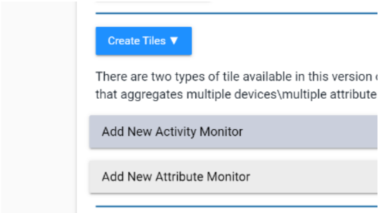
1) The extra money means I can work on developing new Tile Builder modules instead of other money generating activities.
2) The next module will be a multi-device X multi-attribute table which should be especially useful for creating a combined view of key elements.
3) New modules will only be available to registered users.
4) Registered users will get priority for any questions posted on the forum.
5) Help establish a viable market in the Hubitat community for high quality software solutions.

If you don’t want to pay $5 you are still free to use the Standard version as much as you wish.

# Section 2

# Tile Builder Standard

## Creating a Tile Using Activity Monitor



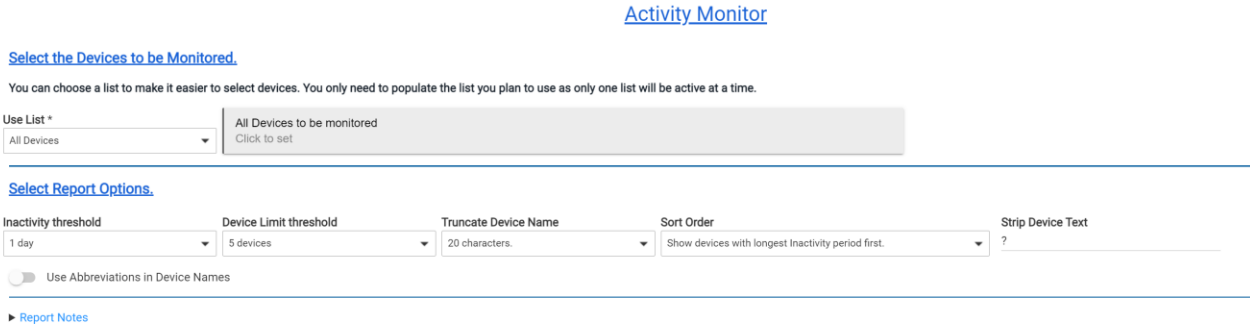
Create Tiles ▼

There are two types of tile available in this version
that aggregates multiple devices\multiple attribute

Add New Activity Monitor

Add New Attribute Monitor

Within the **Tile Builder** parent app go to the section called **Create Tile** and select **Add New Activity Monitor**. The **Activity Monitor** main screen will be displayed. The top part of the display controls which devices will be monitored and how that information will be displayed.



Activity Monitor

Select the Devices to be Monitored.

You can choose a list to make it easier to select devices. You only need to populate the list you plan to use as only one list will be active at a time.

Use List *: All Devices

All Devices to be monitored: Click to set

Select Report Options.

Inactivity threshold: 1 day

Device Limit threshold: 5 devices

Truncate Device Name: 20 characters.

Sort Order: Show devices with longest Inactivity period first.

Strip Device Text: ?

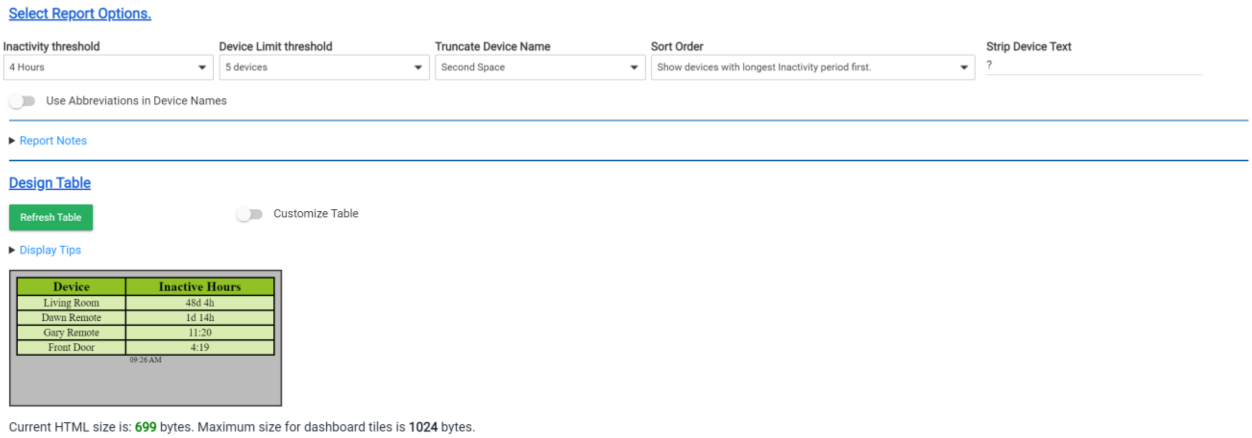
Use Abbreviations in Device Names

▸ Report Notes

We will create a sample looking for inactive battery devices. Note: I have contact sensor devices that fail while the battery still claims to be 100%, so using the percentage alone is not a guarantee of catching a failed battery.

1. On the **Use List** dropdown select **Battery Devices**.
2. Click on the selector now titled **Battery Devices to be Monitored**. This is a filtered list of all your devices that have a battery capability. Select all the devices.
3. The **Inactivity Threshold** will limit the results to only those devices whose inactivity is greater than the threshold value. The default is 1 day, let's change that to 4 hours.
4. The **Device Limit Threshold** limits the display to no more than this number, but it may be less. We will leave that at the default of 5.
5. The **Truncate Device Name** allows you to shorten the device name and is useful in shrinking the amount of data space required, making devices names more consistent in length and to reduce text wrapping. In this example I chose to truncate at the second space.
6. The **Sort Order** allows you to sort the table by oldest or youngest. Given we are looking for devices that might have a failed battery we will set it to **Show devices with longest Inactivity period first**.
7. The **Strip Device Text** allows you to specify a string that will be completely eliminated from the device names. I sometimes use ascii characters such as ! or ~ to group like devices together in the device table and device picker. I can strip those characters from the final display name. We can leave this at the default for this example.
8. The **Use Abbreviations in Device Names** is an option for reducing the size of the data in the result set. With this enabled the word " Room" becomes " Rm", " Sensor" becomes "Sns" etc. When trying to cram a lot of data into a table this can be a very useful option but is mostly of use in **Attribute Monitor** where the result sets are longer. We will leave this turned off here.

Based on these parameters my report looks like this. Only 4 battery devices have not been active in the last 4 hours. The **Living Room** device (a sensor) obviously needs a battery replaced despite the fact the last report stated the battery was at 100%.



Select Report Options.

Inactivity threshold: 4 Hours

Device Limit threshold: 5 devices

Truncate Device Name: Second Space

Sort Order: Show devices with longest Inactivity period first.

Strip Device Text: ?

Use Abbreviations in Device Names

▶ Report Notes

Design Table

Customize Table

▶ Display Tips

| Device | Inactive Hours |
|---|---|
| Living Room | 48d 4h |
| Dawn Remote | 1d 14h |
| Gary Remote | 11:20 |
| Front Door | 4:19 |

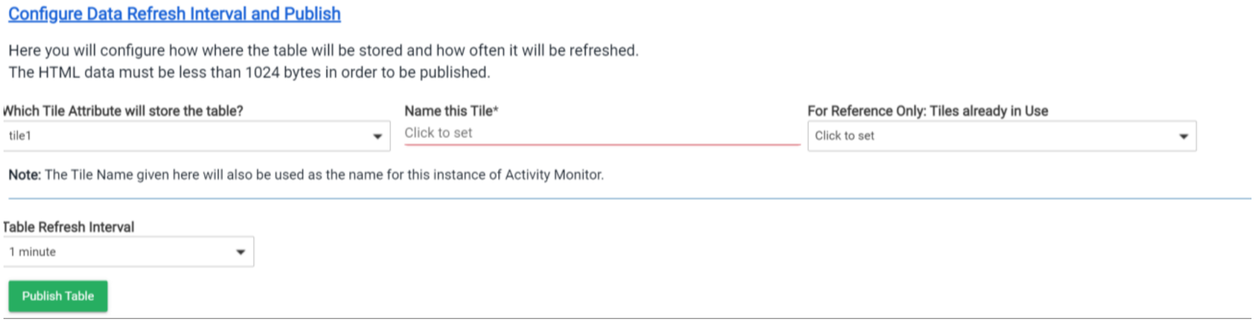
09:26 AM

Current HTML size is: 699 bytes. Maximum size for dashboard tiles is 1024 bytes.

All the table customizations are hidden unless the **Customize Table** option is checked. We will come back to that, but at this point there is nothing else that is absolutely required and we can publish the table as it currently is.

## Publishing a Tile

The publishing options are shown below.



Configure Data Refresh Interval and Publish

Here you will configure how where the table will be stored and how often it will be refreshed.
The HTML data must be less than 1024 bytes in order to be published.

Which Tile Attribute will store the table?: tile1

Name this Tile*: Click to set

For Reference Only: Tiles already in Use: Click to set

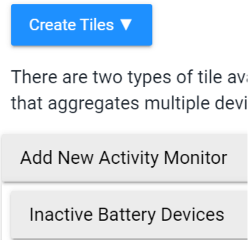
**Note:** The Tile Name given here will also be used as the name for this instance of Activity Monitor.

Table Refresh Interval: 1 minute

To Publish a tile follow these steps:

1. Select the **Tile Attribute** to store the table in. Tile attributes are tile1 – tile 25.  We will use tile1 in this case.
2. **Name the Tile**. I'm going to call it **Inactive Battery Devices – Tile 1**. This is also the name that will be used when looking at the **Tile Builder** parent app. I recommend you append the name with the tile number so you can see it on the parent screen.
3. Select a **Table refresh interval**. No need to make the hub work too hard. In this case I think hourly would be reasonable.
4. With those values set click on **Publish Table**.
5. Click on **Done** to close the **Activity Monitor** app.

6. You should see your new tile listed under the **Activity Monitor** child app like this.

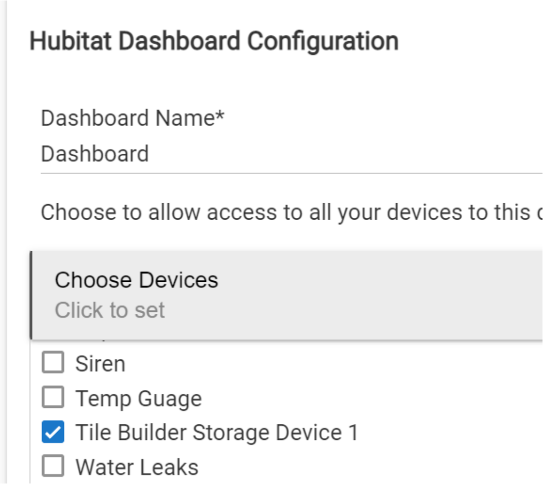


There are two types of tile av
that aggregates multiple devi

You can go back and edit this tile any time by clicking on this button.

## Dashboard Integration

We are done with **Tile Builder** for the moment and now we can add the newly generated tile to the dashboard of our choice. Before we can do that, we must first grant the dashboard access to the **Tile Builder Storage Device** as shown below.



Hubitat Dashboard Configuration

Dashboard Name*
Dashboard

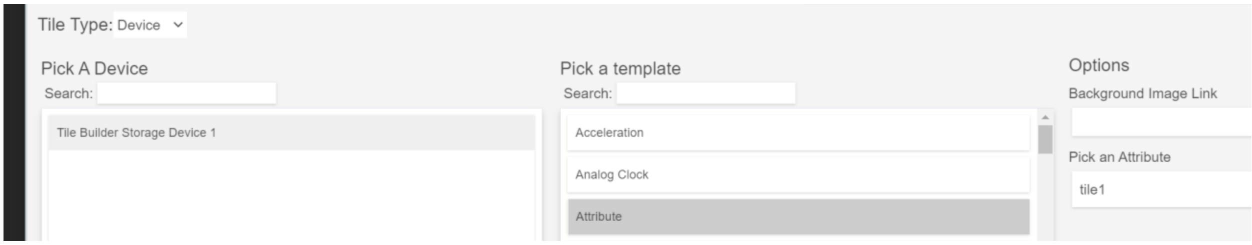
Choose to allow access to all your devices to this d

Choose Devices
Click to set

☐ Siren
☐ Temp Guage
☑ Tile Builder Storage Device 1
☐ Water Leaks

**Note:** Dashboard does not need access to the underlying devices that provide data for the table.

Next, we must add the newly generated tile to the Dashboard in the same way we would add any other device as shown below.



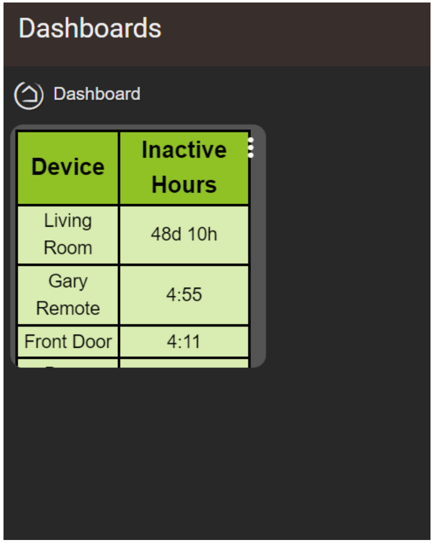
Tile Type: Device

Pick A Device
Search:
Tile Builder Storage Device 1

Pick a template
Search:
Acceleration
Analog Clock
Attribute

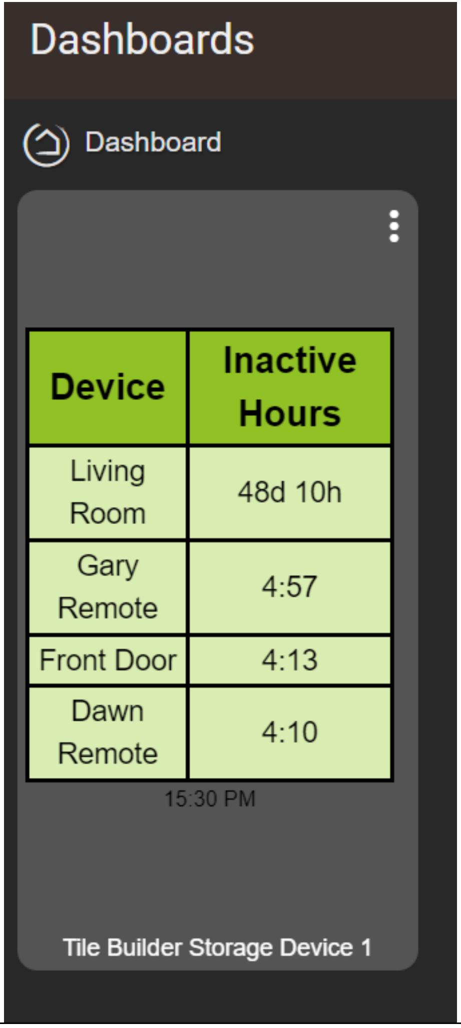
Options
Background Image Link
Pick an Attribute
tile1

The tile will initially look like this.



| Device | Inactive Hours |
|---|---|
| Living Room | 48d 10h |
| Gary Remote | 4:55 |
| Front Door | 4:11 |

By changing the dimensions of a tile to 1 wide x 2 high it will look like this.



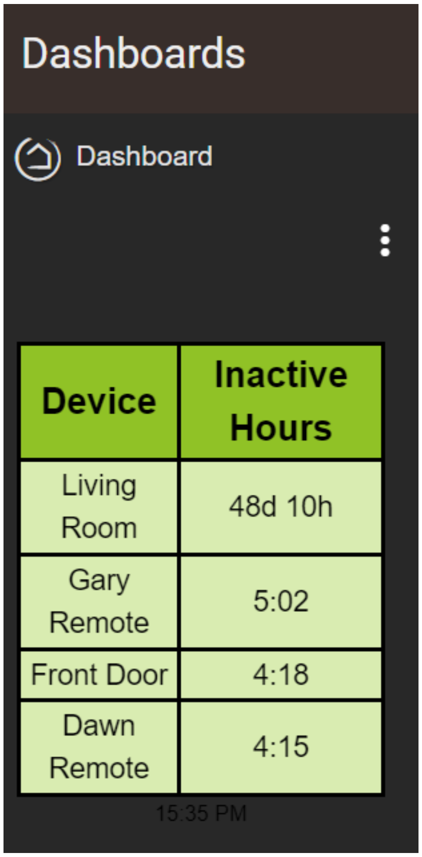
| Device | Inactive Hours |
|---|---|
| Living Room | 48d 10h |
| Gary Remote | 4:57 |
| Front Door | 4:13 |
| Dawn Remote | 4:10 |

If you wish to you can improve the look of the tile by adding these lines to your dashboard CSS.

```
#tile-1 {background-color: rgba(128,128,128,0.0) !important;}
#tile-2 {background-color: rgba(128,128,128,0.0) !important;}
#tile-1 .tile-title, #tile-2 .tile-title {visibility: hidden; display: none;}
```

These make the tile background transparent and then hide the name of the device.
**Note:** The tile number referred to in the above CSS code has no bearing on the tile numbers assigned in the **Tile Builder** app.

With the CSS code applied it is cleaned up a little bit.



| Device | Inactive Hours |
|---|---|
| Living Room | 48d 10h |
| Gary Remote | 5:02 |
| Front Door | 4:18 |
| Dawn Remote | 4:15 |

15:35 PM

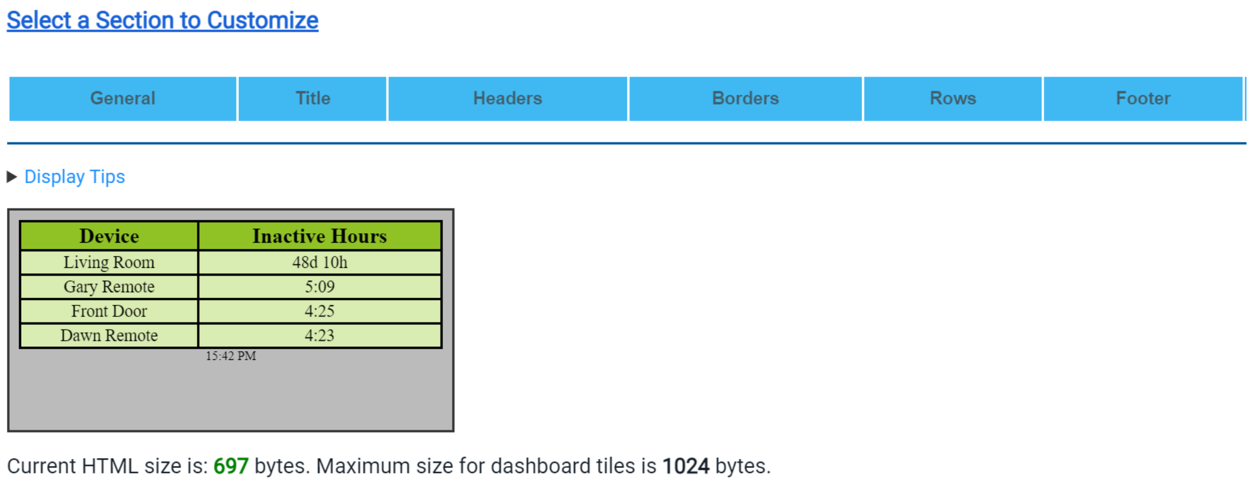
Not bad, it has the basic information but it’s not going to win any design awards. Here are a few things I don’t like about it so far.

1. The font is boring.
2. It’s not clear what the table represents.
3. It is hard to the read timestamp at the bottom of the time. (Last time it was generated.)
4. The table border blends into the background.
5. Green is not my favorite color.

Let us go back and fix these and make it a little more pleasing.

## Editing Activity Monitor Tiles

Open the tile in **Tile Builder** parent app. This time we will click on **Customize Table**. In the standard version it looks like this.



Select a Section to Customize

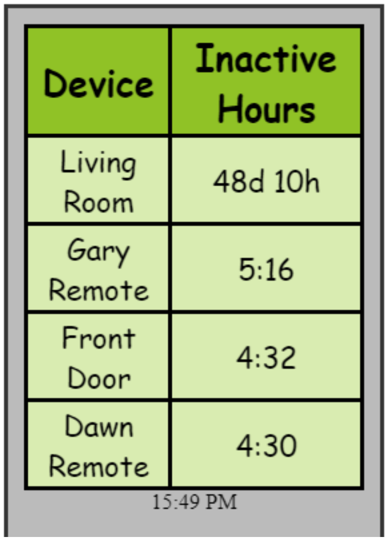
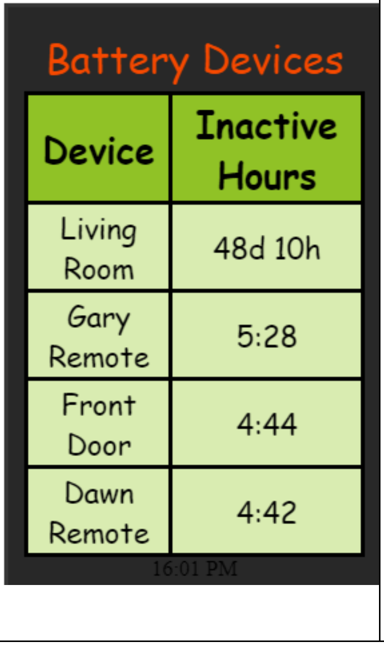
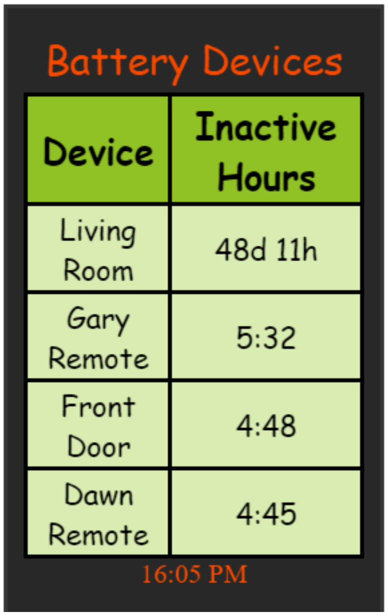
| Device | Inactive Hours |
|---|---|
| Living Room | 48d 10h |
| Gary Remote | 5:09 |
| Front Door | 4:25 |
| Dawn Remote | 4:23 |

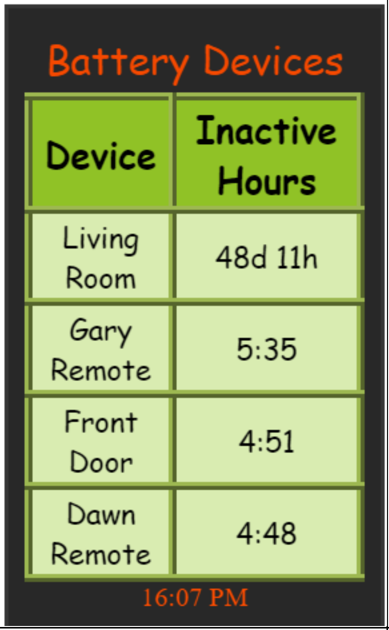
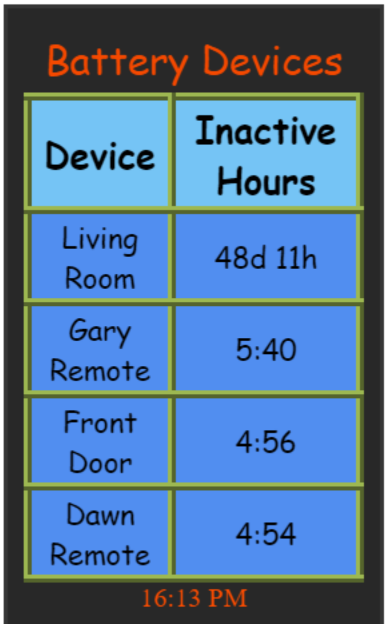
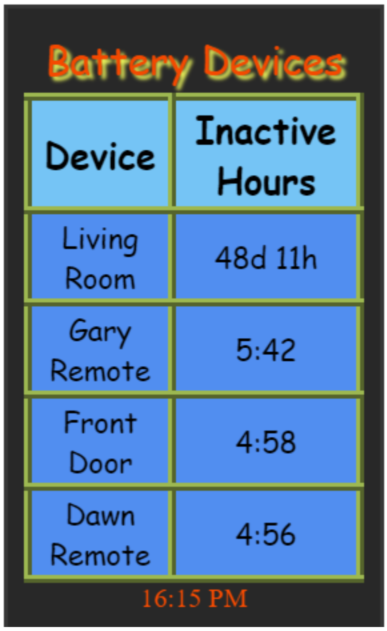
15:42 PM

Current HTML size is: 697 bytes. Maximum size for dashboard tiles is 1024 bytes.

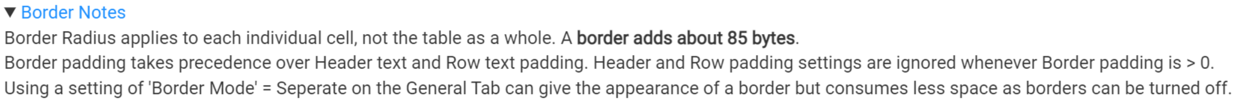
Before we start, let's go to the **General** tab and change the **Select Tile Preview Size** to match our desired size. In this case **1 x 2**. Then click on **Dashboard Color** and use the eye dropper tool to grab the color from your dashboard. This makes the preview more useful, especially when selecting colors.

The following table shows the progression of the table as the different elements are changed.

<table>
<tr>
<td><b>General Tab</b><br>Select font Comic Sans.<br>
<table>
<tr><th>Device</th><th>Inactive Hours</th></tr>
<tr><td>Living Room</td><td>48d 10h</td></tr>
<tr><td>Gary Remote</td><td>5:16</td></tr>
<tr><td>Front Door</td><td>4:32</td></tr>
<tr><td>Dawn Remote</td><td>4:30</td></tr>
</table>
15:49 PM</td>
<td><b>Title Tab - Add Title</b><br>Size 110%, new color.<br>
Battery Devices
<table>
<tr><th>Device</th><th>Inactive Hours</th></tr>
<tr><td>Living Room</td><td>48d 10h</td></tr>
<tr><td>Gary Remote</td><td>5:28</td></tr>
<tr><td>Front Door</td><td>4:44</td></tr>
<tr><td>Dawn Remote</td><td>4:42</td></tr>
</table>
16:01 PM</td>
<td><b>Footer Tab</b><br>Size 75%, new color.<br>
Battery Devices
<table>
<tr><th>Device</th><th>Inactive Hours</th></tr>
<tr><td>Living Room</td><td>48d 11h</td></tr>
<tr><td>Gary Remote</td><td>5:32</td></tr>
<tr><td>Front Door</td><td>4:48</td></tr>
<tr><td>Dawn Remote</td><td>4:45</td></tr>
</table>
16:05 PM</td>
</tr>
</table>

<table>
<tr>
<td><b>Borders Tab – Change Border</b><br>Type: Ridge, Width: 4, new color.<br><br>Battery Devices<table><tr><th>Device</th><th>Inactive Hours</th></tr><tr><td>Living Room</td><td>48d 11h</td></tr><tr><td>Gary Remote</td><td>5:35</td></tr><tr><td>Front Door</td><td>4:51</td></tr><tr><td>Dawn Remote</td><td>4:48</td></tr></table>16:07 PM</td>
<td><b>Header Tab &amp; Row Tab</b><br>Change background color.<br><br>Battery Devices<table><tr><th>Device</th><th>Inactive Hours</th></tr><tr><td>Living Room</td><td>48d 11h</td></tr><tr><td>Gary Remote</td><td>5:40</td></tr><tr><td>Front Door</td><td>4:56</td></tr><tr><td>Dawn Remote</td><td>4:54</td></tr></table>16:13 PM</td>
<td><b>Title Tab</b><br>Add a title shadow.<br><br>Battery Devices<table><tr><th>Device</th><th>Inactive Hours</th></tr><tr><td>Living Room</td><td>48d 11h</td></tr><tr><td>Gary Remote</td><td>5:42</td></tr><tr><td>Front Door</td><td>4:58</td></tr><tr><td>Dawn Remote</td><td>4:56</td></tr></table>16:15 PM</td>
</tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

As you can see it is easy to amend the table and the number of variations is limitless. The purpose of most controls is obvious from the name. Where explanation is needed it is included under a notes section that exists under each tab. The following example is from the **Border** tab.



▼ Border Notes

Border Radius applies to each individual cell, not the table as a whole. A **border adds about 85 bytes**.

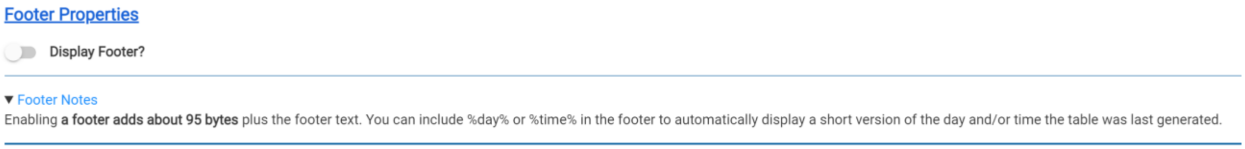
Border padding takes precedence over Header text and Row text padding. Header and Row padding settings are ignored whenever Border padding is > 0.

Using a setting of 'Border Mode' = Seperate on the General Tab can give the appearance of a border but consumes less space as borders can be turned off.

## Tile Size Limits

As previously mentioned, the maximum size of data published to a dashboard via a device attribute is 1,024 bytes. This is quite small so it is helpful to be cognizant of the size of the tile you have created and what components are contributing to that size. **Tile Builder** has capabilities built in to help optimize the tables as well as analyze the usage.

1) Under the **Notes** for each section the added payload from enabling a given option is displayed as shown in this example.



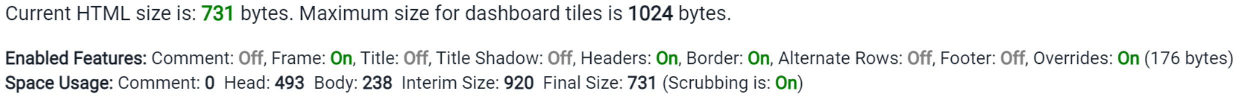
Footer Properties

○ Display Footer?

▼ Footer Notes

Enabling **a footer adds about 95 bytes** plus the footer text. You can include %day% or %time% in the footer to automatically display a short version of the day and/or time the table was last generated.

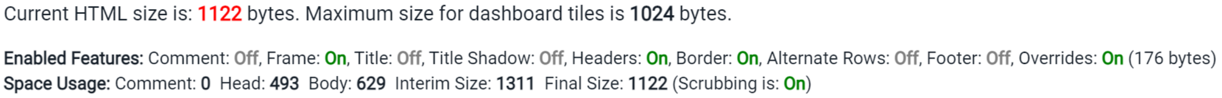
2) The area below the table preview shows the current size of the table, the compressed size of the table and which options are currently enabled. For example:



Current HTML size is: **731** bytes. Maximum size for dashboard tiles is **1024** bytes.

**Enabled Features:** Comment: Off, Frame: **On**, Title: Off, Title Shadow: Off, Headers: **On**, Border: **On**, Alternate Rows: Off, Footer: Off, Overrides: **On** (176 bytes)
**Space Usage:** Comment: **0** Head: **493** Body: **238** Interim Size: **920** Final Size: **731** (Scrubbing is: **On**)

3) **Scrubbing** removes all excess characters from the HTML making it extremely dense and is automatically turned on.
4) When the 1,024 byte limit is exceeded the size will be displayed in red and the **Publish** button will be greyed out (not shown).



Current HTML size is: **1122** bytes. Maximum size for dashboard tiles is **1024** bytes.

**Enabled Features:** Comment: Off, Frame: **On**, Title: Off, Title Shadow: Off, Headers: **On**, Border: **On**, Alternate Rows: Off, Footer: Off, Overrides: **On** (176 bytes)
**Space Usage:** Comment: **0** Head: **493** Body: **629** Interim Size: **1311** Final Size: **1122** (Scrubbing is: **On**)

There are multiple strategies to keep the size of the tile within the limits and these tools will help you find that optimal point more quickly. It is important to note that changing the amount of data in the table is an equally important strategy to sizing tiles appropriately.

**Note:** If you have **Tile Builder Advanced** you can load the style called **Everything Off** which uses all the HTML default values to render the minimal tile size. You can then enable just the components that you deem necessary.

## Creating a Tile Using Attribute Monitor

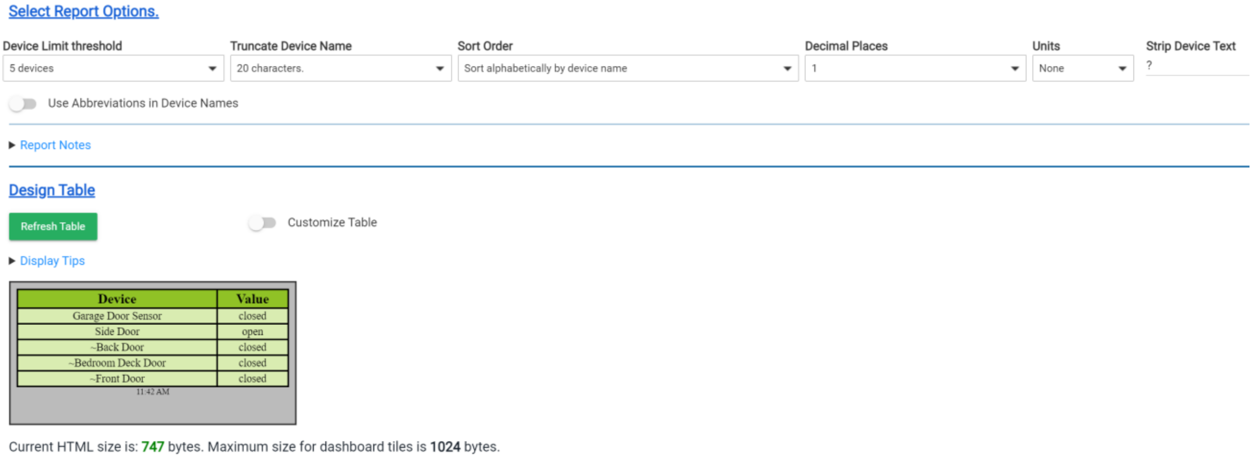
The process for doing this is almost identical to that for creating an **Activity Monitor** tile so I will only focus on the differences.

In **Tile Builder** parent app click on **Add New Attribute Monitor**

We will create a sample to monitor the contact status of doors.

1. On the **Select Attribute to Monitor** dropdown select **Contact**.
2. Click on the **Select Devices to Monitor**. This is a filtered list of all your **Contact** devices. Select all your doors.

In my example the screen now looks like this.



Select Report Options.

| Device Limit threshold | Truncate Device Name | Sort Order | Decimal Places | Units | Strip Device Text |
|---|---|---|---|---|---|
| 5 devices | 20 characters. | Sort alphabetically by device name | 1 | None | ? |

Use Abbreviations in Device Names

▶ Report Notes

Design Table

| Device | Value |
|---|---|
| Garage Door Sensor | closed |
| Side Door | open |
| ~Back Door | closed |
| ~Bedroom Deck Door | closed |
| ~Front Door | closed |

11:42 AM

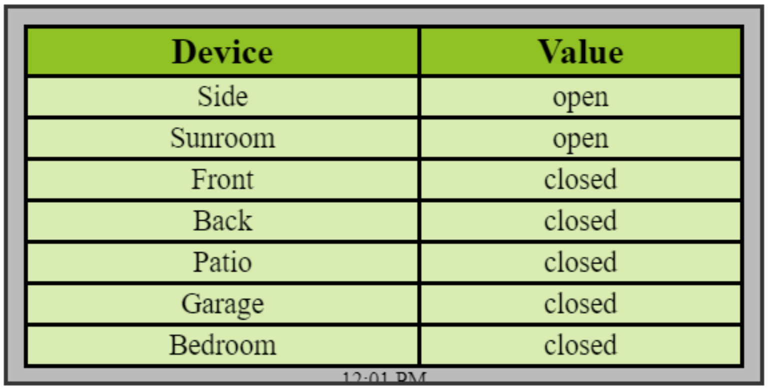
Current HTML size is: 747 bytes. Maximum size for dashboard tiles is 1024 bytes.

Let us go through each of the report options.

1. **Device Limit Threshold**: You can limit the number of devices displayed to be less than the number of devices being monitored. For example, I might monitor 12 temperatures but only display the 5 highest ones by using the appropriate sort order.
2. **Truncate Device Name**: Allows you to shorten the device name and is useful in shrinking the amount of data space required, making devices names more consistent in length and to reduce text wrapping. In this example I chose to truncate at the first space because I don't need the word door to be on every line.
3. **Sort Order**: Allows you to sort the table by device name, value, or value reverse. I care more about open doors so I will have them sort first by selecting the **Reverse Sort Alphabetically by Value** option.
4. **Decimal Places**: Applies only to floating point values.
5. **Units**: You can choose the type of units to display alongside the value. Not relevant here.

6. **Strip Device Text**: Allows you to specify a character or string that will be eliminated from the device names display. I sometimes use ascii characters such as ! or ~ to group like devices together in the device table and device picker. In this case my wired contacts have a ~ as the first character and it looks unsightly. I can strip those characters from the final display name by entering the ~ character in this field.
7. The **Use Abbreviations in Device Names** is an option for reducing the size of the data in the result set. With this enabled the word " Room" becomes " Rm", " Sensor" becomes "Sns" etc. We will leave this turned off here as we have truncated device names at the first word, so they are pretty short already.

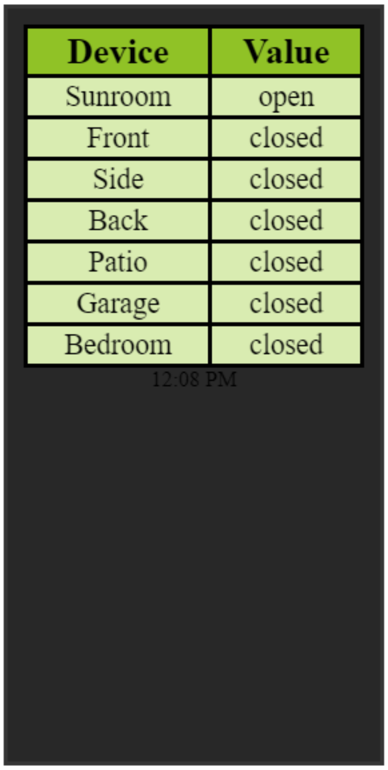
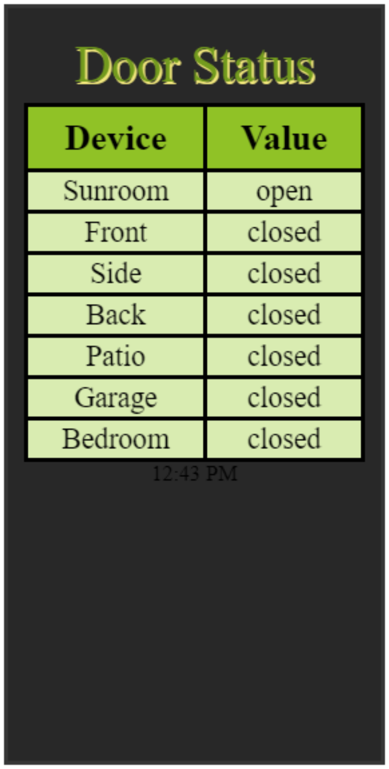
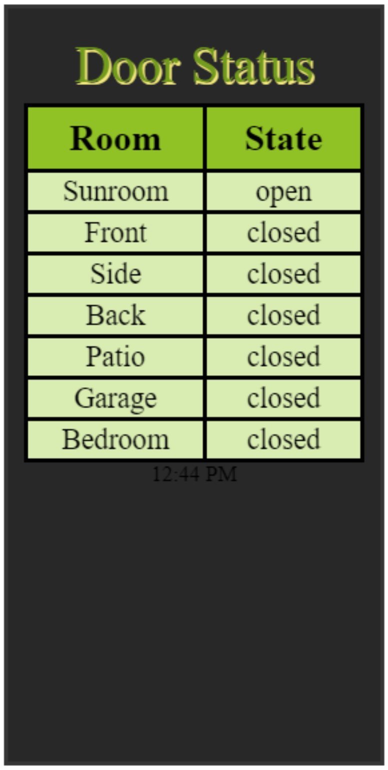
Our table now looks like this.

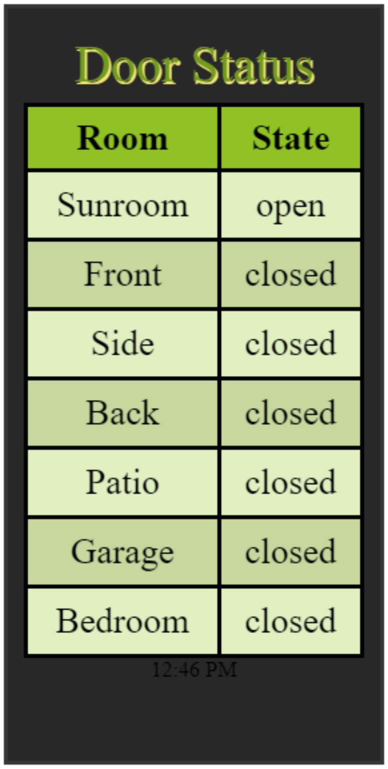
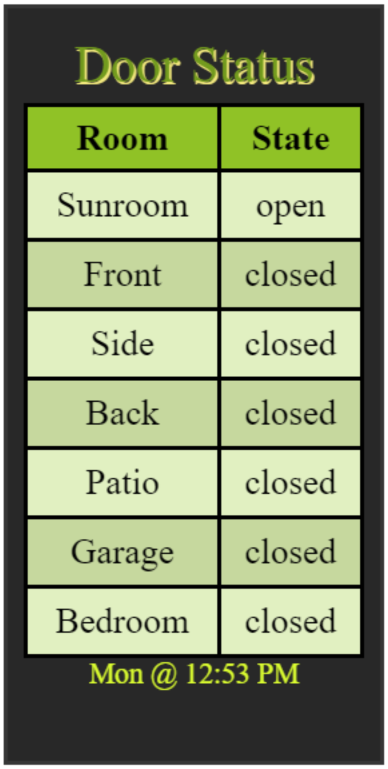


| Device | Value |
|---|---|
| Side | open |
| Sunroom | open |
| Front | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

12:01 PM

As you can see the table is tight, the footer is getting cut off and I would like to add a title to it so I'm going to go to the General tab and change the tile preview to **1x2** as I think that will be a better fit. I also changed the Dashboard Color to match my actual dashboard to give a more accurate preview. The table below shows the progression of the tile preview as I make each of the changes listed.

<table>
<tr>
<td><b>General Tab</b><br>Tile preview <b>1x2</b>, Dashboard color.<br>

<table>
<tr><th>Device</th><th>Value</th></tr>
<tr><td>Sunroom</td><td>open</td></tr>
<tr><td>Front</td><td>closed</td></tr>
<tr><td>Side</td><td>closed</td></tr>
<tr><td>Back</td><td>closed</td></tr>
<tr><td>Patio</td><td>closed</td></tr>
<tr><td>Garage</td><td>closed</td></tr>
<tr><td>Bedroom</td><td>closed</td></tr>
</table>
12:08 PM</td>
<td><b>Title Tab</b><br>Add Title, color, padding, shadow.<br>

Door Status
<table>
<tr><th>Device</th><th>Value</th></tr>
<tr><td>Sunroom</td><td>open</td></tr>
<tr><td>Front</td><td>closed</td></tr>
<tr><td>Side</td><td>closed</td></tr>
<tr><td>Back</td><td>closed</td></tr>
<tr><td>Patio</td><td>closed</td></tr>
<tr><td>Garage</td><td>closed</td></tr>
<tr><td>Bedroom</td><td>closed</td></tr>
</table>
12:43 PM</td>
<td><b>Headers Tab</b><br>Header text<br>

Door Status
<table>
<tr><th>Room</th><th>State</th></tr>
<tr><td>Sunroom</td><td>open</td></tr>
<tr><td>Front</td><td>closed</td></tr>
<tr><td>Side</td><td>closed</td></tr>
<tr><td>Back</td><td>closed</td></tr>
<tr><td>Patio</td><td>closed</td></tr>
<tr><td>Garage</td><td>closed</td></tr>
<tr><td>Bedroom</td><td>closed</td></tr>
</table>
12:44 PM</td>
</tr>
<tr>
<td><b>Rows Tab</b><br>Text padding, alternate row color, background opacity.</td>
<td><b>Footer Tab</b><br>Footer text %day% @ %time%, text color.</td>
<td></td>
</tr>
</table>

| | | |
|---|---|---|
|  |  | |

Door Status

| Room | State |
|---|---|
| Sunroom | open |
| Front | closed |
| Side | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

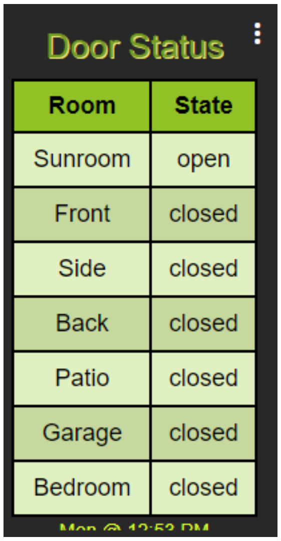
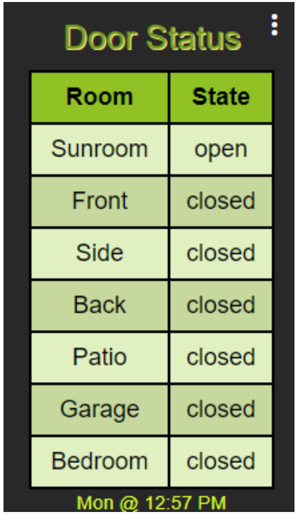
12:46 PM

Door Status

| Room | State |
|---|---|
| Sunroom | open |
| Front | closed |
| Side | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

Mon @ 12:53 PM

When I first published the result to the dashboard it looked like the entry in the **before** column. Notice how the footer is partially cut off. This demonstrates that while the preview is a close approximation it is not exact.  It's an easy fix, I went back to the **Rows tab** and changed the **Text Padding** from 6 to 5 and then everything displayed as expected as shown in the **After** column..

| **Before** | **After changes to row text padding.** |
|---|---|
|  |  |

Door Status

| Room | State |
|---|---|
| Sunroom | open |
| Front | closed |
| Side | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

Mon @ 12:53 PM

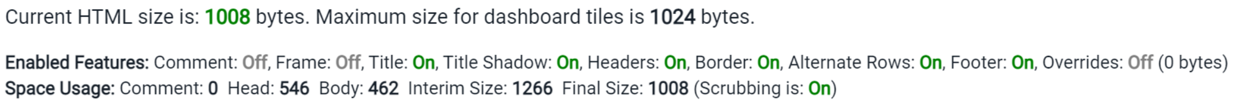
Door Status

| Room | State |
|---|---|
| Sunroom | open |
| Front | closed |
| Side | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

Mon @ 12:57 PM

## Tile Size Revisited

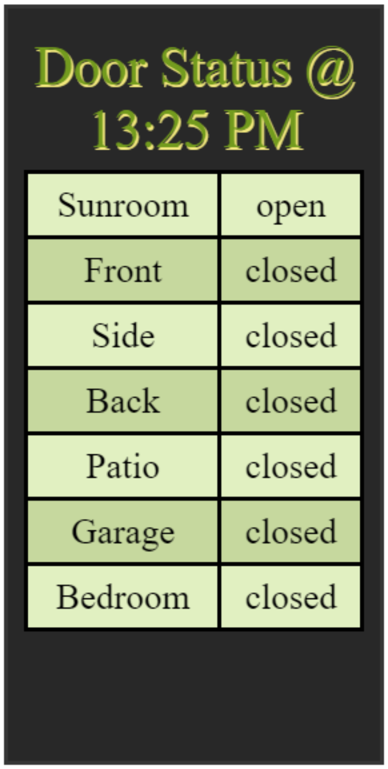
In this case the final tile size was 1,008 bytes after scrubbing and close to the upper limit of 1,024. As a closed door uses 2 bytes more than an **open** door, we need to leave a some buffer space to handle the resulting variations in tile size. In this case the less frequent state (open) is shorter than the more frequent state (closed) so this is not likely to be an issue in this case.



Current HTML size is: **1008** bytes. Maximum size for dashboard tiles is **1024** bytes.

**Enabled Features:** Comment: Off, Frame: Off, Title: **On**, Title Shadow: **On**, Headers: **On**, Border: **On**, Alternate Rows: **On**, Footer: **On**, Overrides: Off (0 bytes)
**Space Usage:** Comment: **0** Head: **546** Body: **462** Interim Size: **1266** Final Size: **1008** (Scrubbing is: **On**)

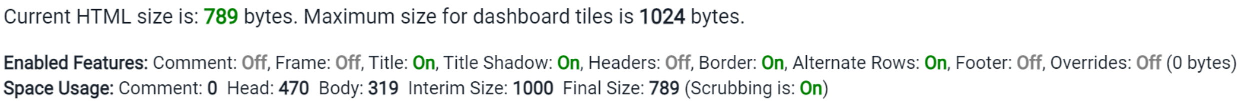
If we wished to add more lines of data, we would have to cut some of the features that were turned on as shown above. In this case I would think about turning off the header rows as the title tells me what the rest of the table is about. I would also consider moving the footer text into the Title. If I do both of those things the tile now looks like this:



Door Status @ 13:25 PM

| | |
|---|---|
| Sunroom | open |
| Front | closed |
| Side | closed |
| Back | closed |
| Patio | closed |
| Garage | closed |
| Bedroom | closed |

If we look at the HTML info, we can see the tile size has shrunk from 1,008 bytes to 789. That is a substantial difference and would be sufficient to add another 5-10 rows of data if desired.



Current HTML size is: **789** bytes. Maximum size for dashboard tiles is **1024** bytes.

**Enabled Features:** Comment: Off, Frame: Off, Title: **On**, Title Shadow: **On**, Headers: Off, Border: **On**, Alternate Rows: **On**, Footer: Off, Overrides: Off (0 bytes)
**Space Usage:** Comment: **0** Head: **470** Body: **319** Interim Size: **1000** Final Size: **789** (Scrubbing is: **On**)

## Embedded HTML Tags

Anywhere you can enter text you can wrap it inside HTML tags. But rather than using **<>** you must use **[]** as normal html tags are rejected by the Hubitat interface. For example, you could enter **[u]Door Status[/u]** in the title field and the title would be displayed in underline, **[b]Door Status[/b]** and it would display in bold. Multiple html tags can be used if they follow html conventions.

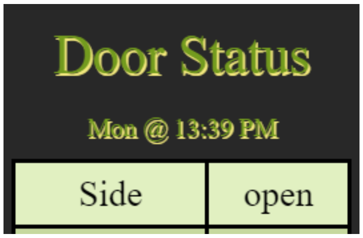
## Macros

The following values are macros that will be expanded in the final html.

- **%day%** will be replaced by a short version of the day name.
- **%time%** will be replaced by a 24 hr time including AM\PM.
- **%units%** will be expanded into the Units chosen, if any.

In the above example the **Door Status** title text was replaced with **Door Status @ %time%.**

Using the text **Door Status [br][font size=2]%day% @ %time%[font]** as the title would look like this.



Door Status

Mon @ 13:39 PM

| | |
|---|---|
| Side | open |

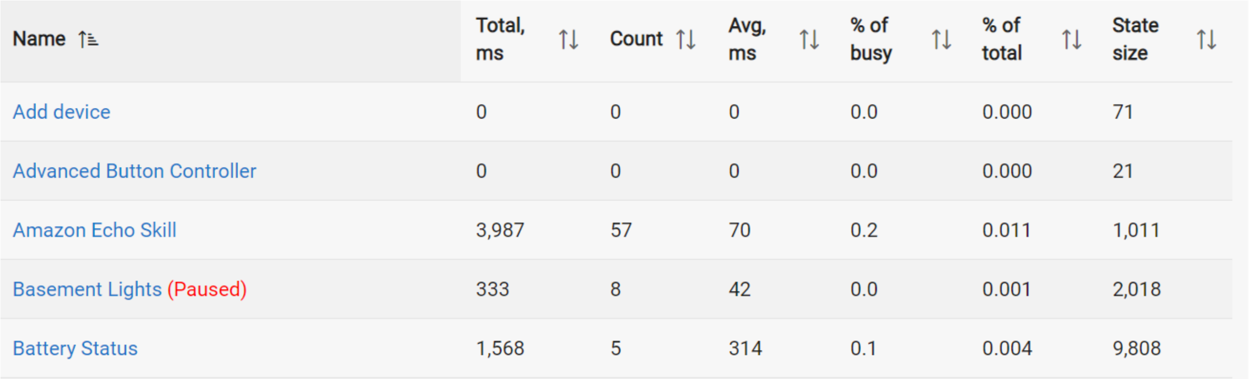
You can use these macros in any text field combined with html tags.

## Publish and Subscribe Model

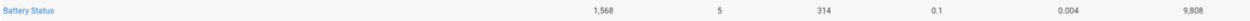
When you click on the **Publish and Subscribe** button, **Tile Builder** creates an event subscription to each of the selected devices and chosen attribute. **Attribute Builder** then remains dormant until such time as one of the monitored attributes changes, in the above case if a door opens or closes, at which point the table is immediately regenerated and published. This is a highly efficient model, and **Tile Builder** tiles will only regenerate when the underlying data has changed. Tile updates will appear on the dashboard at the same speed as a typical device tile.

# Hubitat CPU Utilization

You can view the performance of **Tile Builder** apps under **Logs\App Stats**. In the below example you can see that the **Battery Status** has only been initiated 5 times since the last reboot (reboots nightly) and has consumed a total of 0.004% of total CPU. This is expected as battery percentage changes do not occur often.



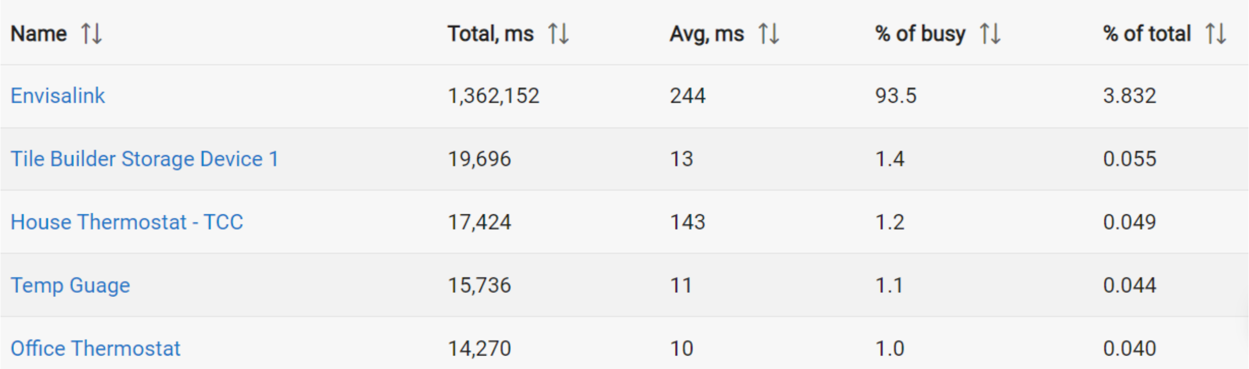
| Name | Total, ms | Count | Avg, ms | % of busy | % of total | State size |
|---|---|---|---|---|---|---|
| Add device | 0 | 0 | 0 | 0.0 | 0.000 | 71 |
| Advanced Button Controller | 0 | 0 | 0 | 0.0 | 0.000 | 21 |
| Amazon Echo Skill | 3,987 | 57 | 70 | 0.2 | 0.011 | 1,011 |
| Basement Lights (Paused) | 333 | 8 | 42 | 0.0 | 0.001 | 2,018 |
| Battery Status | 1,568 | 5 | 314 | 0.1 | 0.004 | 9,808 |



Battery Status 1,568 5 314 0.1 0.004 9,808

In addition to the **Tile Builder** apps utilizing the CPU the larger consumer is the **Tile Builder Storage Driver**. You can track this in Logs\Device Stats as shown below.

For the instance shown below there are 8 published tiles, and the **Tile Builder Storage Device** is about as busy as a thermostat.

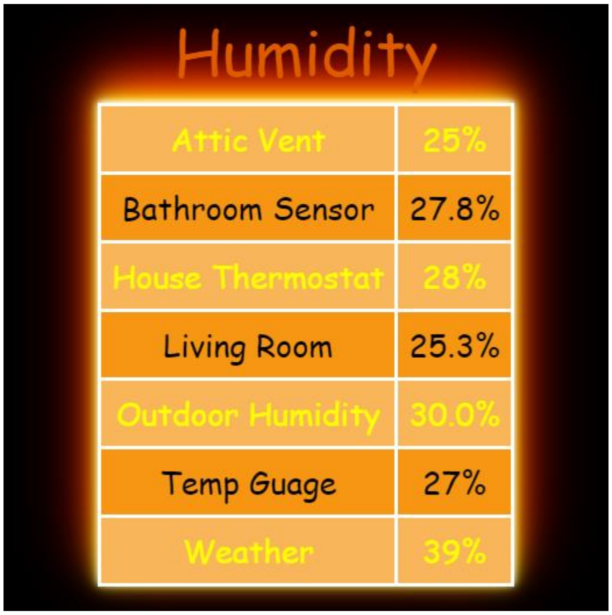
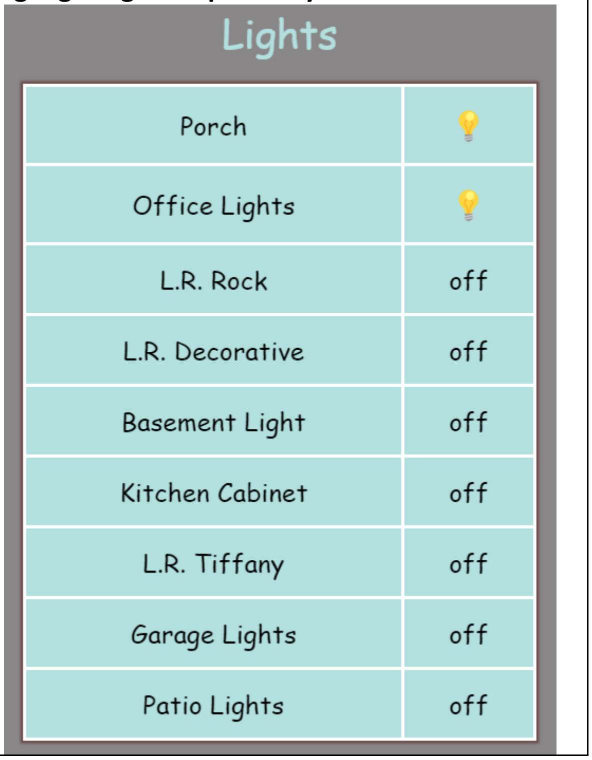


| Name | Total, ms | Avg, ms | % of busy | % of total |
|---|---|---|---|---|
| Envisalink | 1,362,152 | 244 | 93.5 | 3.832 |
| Tile Builder Storage Device 1 | 19,696 | 13 | 1.4 | 0.055 |
| House Thermostat - TCC | 17,424 | 143 | 1.2 | 0.049 |
| Temp Guage | 15,736 | 11 | 1.1 | 0.044 |
| Office Thermostat | 14,270 | 10 | 1.0 | 0.040 |

Final CPU utilization in any given case will be a function of the number of tiles published and especially how frequently those tiles are updated. For this reason, publishing rapidly changing data such as power consumption needs to be done thoughtfully.

## Summary

As you can see, **Tile Builder** offers a great deal of flexibility in how to display data onto a Hubitat dashboard. What you have learned so far should keep you busy for a while. While **Tile Builder** has a lot to offer, **Tile Builder Advanced** is a big step up in terms of flexibility and control. The tiles in the table shown below were created with **Tile Builder Advanced**.

| **Style Example - Halloween** | **Highlighting Example – Keyword substitution** |
|---|---|
|  |  |

Humidity

| | |
|---|---|
| Attic Vent | 25% |
| Bathroom Sensor | 27.8% |
| House Thermostat | 28% |
| Living Room | 25.3% |
| Outdoor Humidity | 30.0% |
| Temp Guage | 27% |
| Weather | 39% |

Lights

| | |
|---|---|
| Porch | 💡 |
| Office Lights | 💡 |
| L.R. Rock | off |
| L.R. Decorative | off |
| Basement Light | off |
| Kitchen Cabinet | off |
| L.R. Tiffany | off |
| Garage Lights | off |
| Patio Lights | off |

In the next section we will explore all the capabilities added to the **Tile Builder Advanced** version.

# Section 3

# Tile Builder Advanced

## Overview of Advanced Features

**Tile Builder Advanced** adds significant capabilities to the construction of tiles. In summary these are as follows:

- **Highlighting**

  Highlighting is a way to draw attention to specific values that are of greater importance by changing how those values look and making them more prominent.

  **Keywords:** These are used to match a string value and enhance those with color, size or completely replace the value. For example, rather than display the word **closed** for a contact sensor you could specify ✓ or **O.K.** be used instead.

  **Thresholds:** These allow numeric values that meet >=, ==, or <= conditions to be highlighted or replaced with a text value. **Batteries <= 50%** could all display as Low instead.

- **Styles**

  Styles are a collection of settings for quickly and consistently modifying how data is displayed.

  **Built-In Styles: Tile Builder** has over a dozen Styles built into the app. These are there primarily to spark your imagination and demonstrate the power of Styles. These are not polished color guides but I expect to add more of those later.

  **Apply Styles**: Apply a built-in style or your own style to a table. This only affects those properties associated with how the table looks and feels. It does not affect the contents of fields such as Title, Headers, or Footer.

  **Save Styles**: Create your own styles and save them for future use.

  **Delete Style**: Remove a style you no longer wish to keep.

  **Import\Export**: Share style strings with others via **Hubitat Community** forums or save an imported style string as a new Style.

- **Advanced\Overrides**

  **Enable Overrides**: **Overrides** are a powerful tool akin to a programming model. With **overrides** you can achieve all kinds of visual effects on a table by overriding or adding to the existing contents of a field. This allows for the creation of new classes and have those classes act on various parts of the table such as the header or row data only.

  **Show Effective Settings**: A diagnostic tool for showing the combined result of normal and override settings.

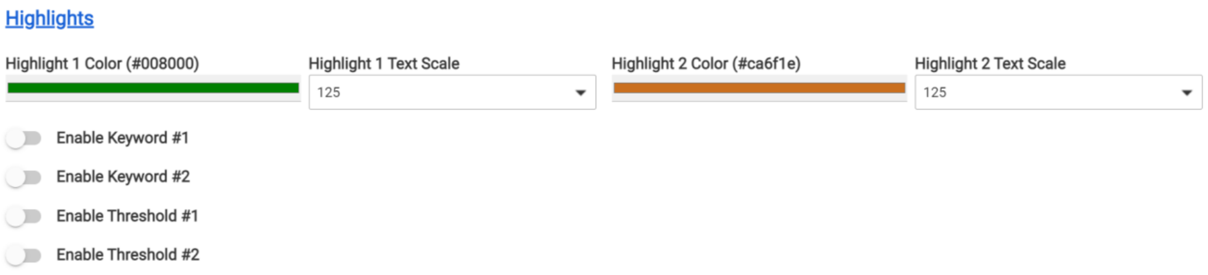
  **Show Pseudo HTML**: A diagnostic tool for showing the final HTML but using **[]** instead of **<>**.

  **Overrides Helper**: A collection of 40+ examples of **overrides** used to achieve all kinds of style effects not possible through the interface.

We will go through each of these sections in more detail in the remainder of this section.

## Highlighting

The Highlights interface looks like this and is only available in **Attribute Monitor**.



Highlights

| Highlight 1 Color (#008000) | Highlight 1 Text Scale | Highlight 2 Color (#ca6f1e) | Highlight 2 Text Scale |
|---|---|---|---|
| 🟩 | 125 | 🟧 | 125 |

○ Enable Keyword #1

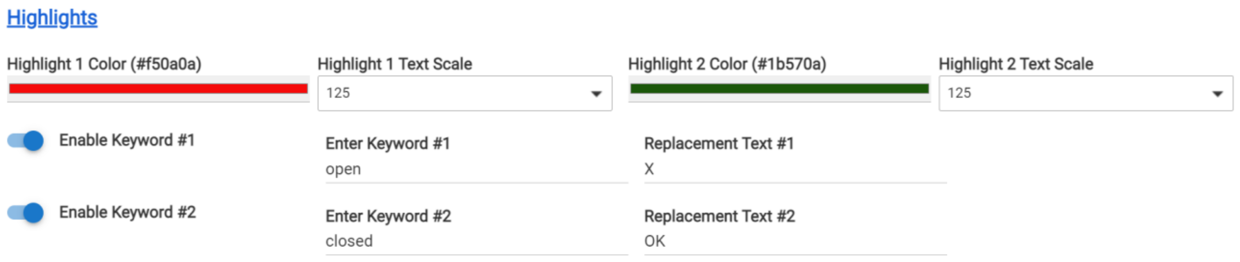
○ Enable Keyword #2

○ Enable Threshold #1

○ Enable Threshold #2

### Keywords

This is a remarkably simple concept, simply stated you can replace one string with another to produce a more attractive result. The replacement string\character can be animated, see the **Classes** section later for details on this. **Note:** When increasing a highlight scale choosing a scale greater than 100% will cause the height of the row to change slightly to accommodate it.

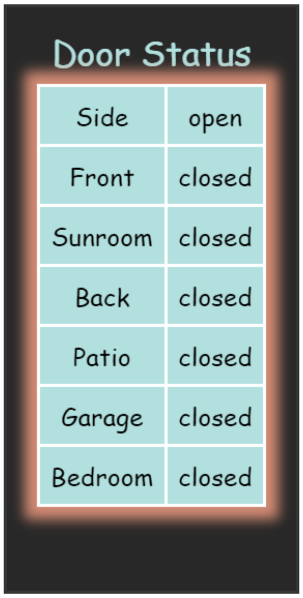
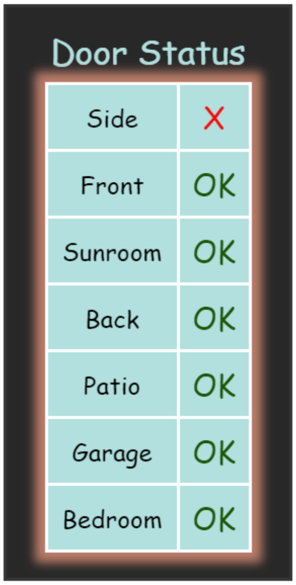


Highlights

| Highlight 1 Color (#f50a0a) | Highlight 1 Text Scale | Highlight 2 Color (#1b570a) | Highlight 2 Text Scale |
|---|---|---|---|
| 🟥 | 125 | 🟩 | 125 |

● Enable Keyword #1 — Enter Keyword #1: open — Replacement Text #1: X

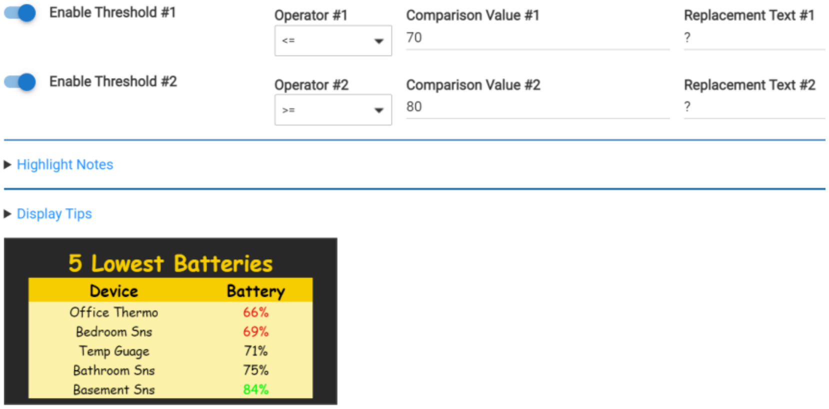
● Enable Keyword #2 — Enter Keyword #2: closed — Replacement Text #2: OK

| Without Highlighting | With Highlighting |
|---|---|
|  |  |
| Door Status<br>Side open<br>Front closed<br>Sunroom closed<br>Back closed<br>Patio closed<br>Garage closed<br>Bedroom closed | Door Status<br>Side X<br>Front OK<br>Sunroom OK<br>Back OK<br>Patio OK<br>Garage OK<br>Bedroom OK |
| **Size: 847 bytes** | **Size: 979 bytes** |

Each active highlight style adds 35 bytes, plus 11 bytes per affected row to the HTML size. In the above example we added 130 bytes in total. If this was too high, we could just enable the keyword for **open** and ignore **closed**. The space required for this option would be 891 bytes but would increase as multiple doors were opened.

## Thresholds

Thresholds are like keywords but act on numerical data vs. strings. Available comparisons are <=, ==, and >= . The example below uses two of these comparisons to stratify the battery levels. A match on **Threshold 1** will apply the **Highlight 1** color to the result, likewise for **Threshold 2**.



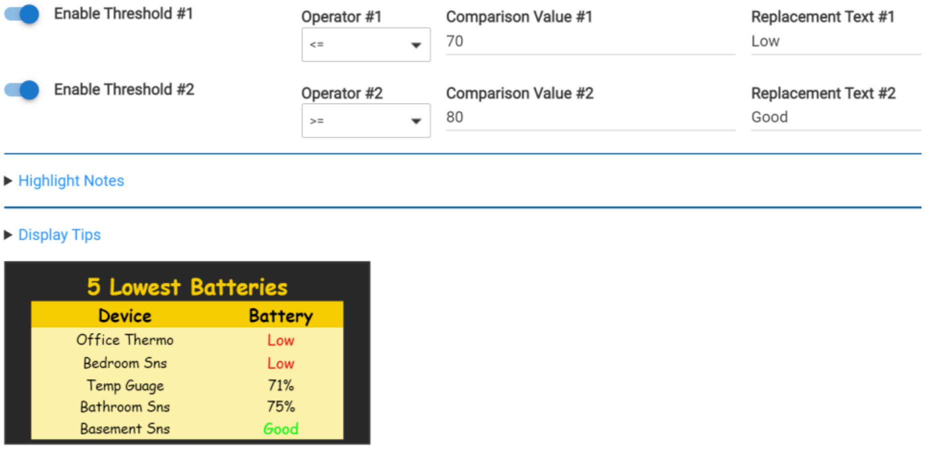
Enable Threshold #1 — Operator #1: <= — Comparison Value #1: 70 — Replacement Text #1: ?

Enable Threshold #2 — Operator #2: >= — Comparison Value #2: 80 — Replacement Text #2: ?

5 Lowest Batteries

| Device | Battery |
|---|---|
| Office Thermo | 66% |
| Bedroom Sns | 69% |
| Temp Guage | 71% |
| Bathroom Sns | 75% |
| Basement Sns | 84% |

Numeric values meeting threshold conditions can also be replaced by text values as shown here.



Enable Threshold #1 — Operator #1: <= — Comparison Value #1: 70 — Replacement Text #1: Low

Enable Threshold #2 — Operator #2: >= — Comparison Value #2: 80 — Replacement Text #2: Good

5 Lowest Batteries

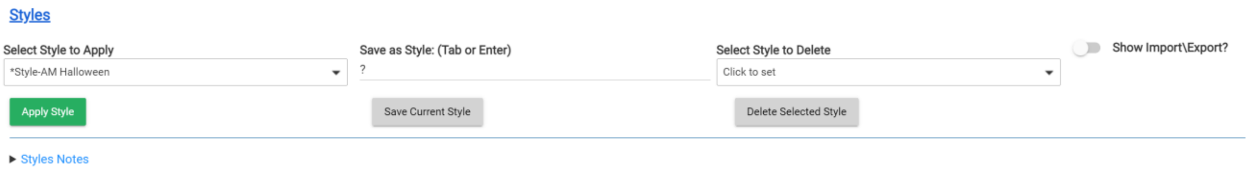
| Device | Battery |
|---|---|
| Office Thermo | Low |
| Bedroom Sns | Low |
| Temp Guage | 71% |
| Bathroom Sns | 75% |
| Basement Sns | Good |

## HTML Tags

You can use embedded HTML within the text replacement fields by using **[]** braces. For example, replacing **closed** with '**[b]O.K.[/b]**' will make the result show in bold as '**O.K.**' which has a bit more punch.

## Styles

The options under the Style tab look like this.



Styles

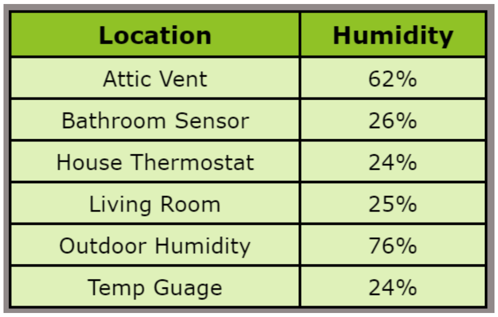
Select Style to Apply: *Style-AM Halloween

Save as Style: (Tab or Enter) ?

Select Style to Delete: Click to set

Show Import\Export?

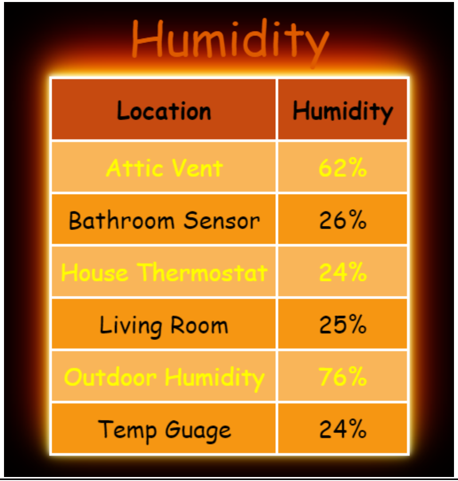
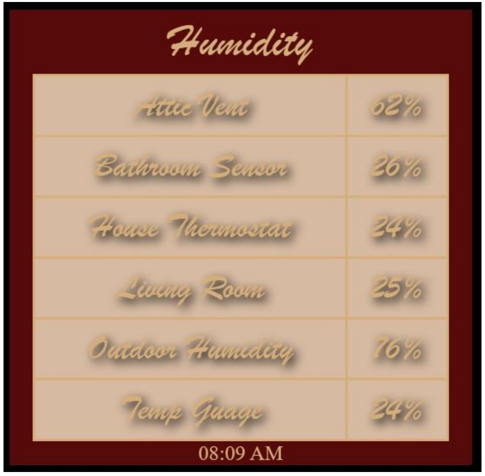
Styles are a quick way of grouping all table settings (not data) together in a convenient package. You may generate an initial table that looks like this.



| Location | Humidity |
|---|---|
| Attic Vent | 62% |
| Bathroom Sensor | 26% |
| House Thermostat | 24% |
| Living Room | 25% |
| Outdoor Humidity | 76% |
| Temp Guage | 24% |

### Apply Style

By applying a style, you can effortlessly make a tile look entirely different.

| Style: Halloween | Style: Wood |
|---|---|
|  |  |

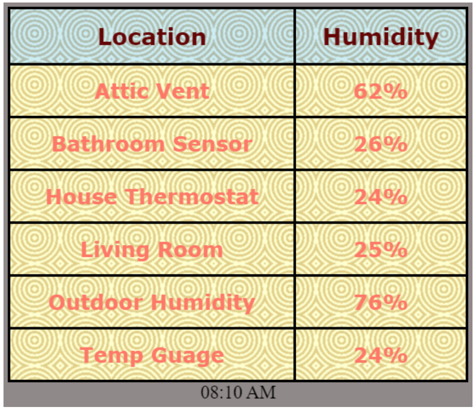
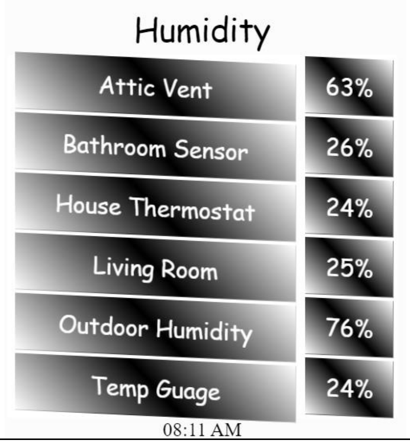
Humidity

| Location | Humidity |
|---|---|
| Attic Vent | 62% |
| Bathroom Sensor | 26% |
| House Thermostat | 24% |
| Living Room | 25% |
| Outdoor Humidity | 76% |
| Temp Guage | 24% |

Humidity

| | |
|---|---|
| Attic Vent | 62% |
| Bathroom Sensor | 26% |
| House Thermostat | 24% |
| Living Room | 25% |
| Outdoor Humidity | 76% |
| Temp Guage | 24% |

08:09 AM

| Style: Pastel Swirl | Style: Black and White |
|---|---|
|  |  |

| Location | Humidity |
|---|---|
| Attic Vent | 62% |
| Bathroom Sensor | 26% |
| House Thermostat | 24% |
| Living Room | 25% |
| Outdoor Humidity | 76% |
| Temp Guage | 24% |

08:10 AM

Humidity

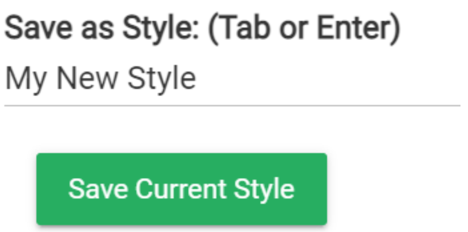
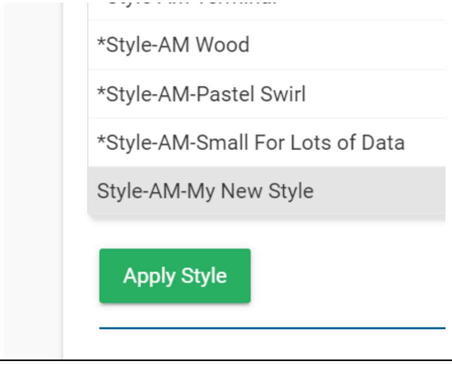
| | |
|---|---|
| Attic Vent | 63% |
| Bathroom Sensor | 26% |
| House Thermostat | 24% |
| Living Room | 25% |
| Outdoor Humidity | 76% |
| Temp Guage | 24% |

08:11 AM

Size is always a consideration given the 1,024-byte limit. The more elements in a style, the larger it will be. All the prior examples are in the 900-950 byte range and could easily be published as shown.

## Save Style

We can create our own styles so that we can easily make numerous tables consistent without having to manually apply the same settings over and over. Once you have the table looking as you wish, simply type the new style name into the text box. Use Tab or Enter to submit that change and the **Save Current Style** button will no longer be greyed out and can be clicked.

| Style: Save New Style | Style: Now available to apply. |
|---|---|
|  |  |

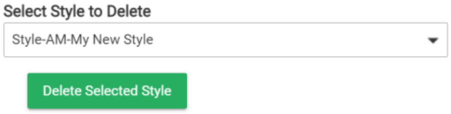
Save as Style: (Tab or Enter)

My New Style

Built-In styles are preceded with an * so they sort to the top of the list. User created styles will appear at the bottom of the list in alphabetical order. Styles are stored under the parent app and are shareable between **Activity Monitor** and **Attribute Monitor**. Future modules will share these styles where the color scheme is compatible.

## Delete Style

To delete a style simply select it from the list and click **Deleted Selected Style**.



Select Style to Delete

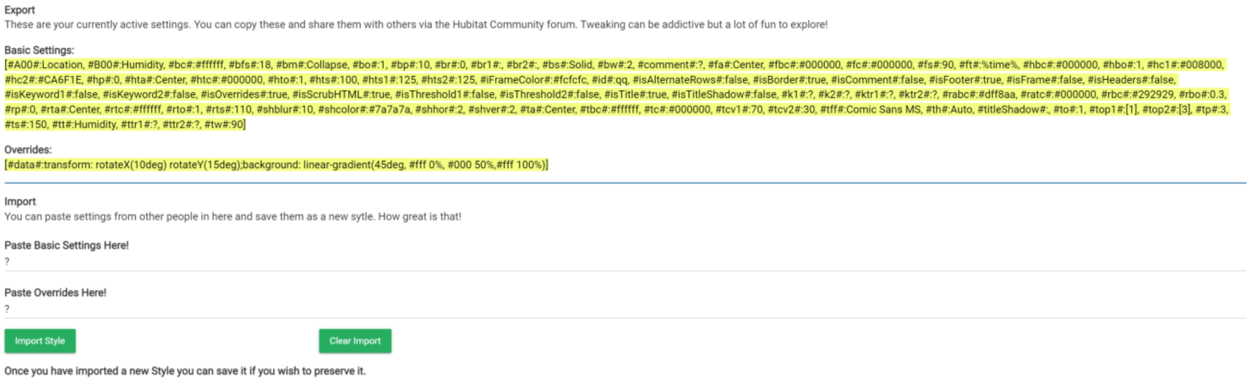
Style-AM-My New Style

## Import\Export

**Hubitat** has a vibrant community forum and it is a great asset, so why not use it. When **Tile Builder Advanced** users produce an attractive or innovative style, they can easily share it with others via the forum using the **Import\Export** functions.

### Export a Style

The screen below shows the **Export** settings for the currently active configuration split into **Basic Settings** and **Overrides**. Don’t worry if these strings do not make any sense yet, they will after you understand **Overrides** which are covered later.



Export

These are your currently active settings. You can copy these and share them with others via the Hubitat Community forum. Tweaking can be addictive but a lot of fun to explore!

Basic Settings:

```
[#A00#:Location, #B00#:Humidity, #bc#:#ffffff, #bfs#:18, #bm#:Collapse, #bo#:1, #bp#:10, #br#:0, #br1#:, #br2#:, #bs#:Solid, #bw#:2, #comment#:?, #fa#:Center, #fbc#:#000000, #fc#:#000000, #fs#:90, #ft#:%time%, #hbc#:#000000, #hbo#:1, #hc1#:#008000,
#hc2#:#CA6F1E, #hp#:0, #hta#:Center, #htc#:#000000, #hto#:1, #hts#:100, #hts1#:125, #hts2#:125, #iFrameColor#:#fcfcfc, #id#:qq, #isAlternateRows#:false, #isBorder#:true, #isComment#:false, #isFooter#:true, #isFrame#:false, #isHeaders#:false,
#isKeyword1#:false, #isKeyword2#:false, #isOverrides#:true, #isScrubHTML#:true, #isThreshold1#:false, #isThreshold2#:false, #isTitle#:true, #isTitleShadow#:false, #k1#:?, #k2#:?, #ktr1#:?, #ktr2#:?, #rabc#:#dff8aa, #ratc#:#000000, #rbc#:#292929, #rbo#:0.3,
#rp#:0, #rta#:Center, #rtc#:#ffffff, #rto#:1, #rts#:110, #shblur#:10, #shcolor#:#7a7a7a, #shhor#:2, #shver#:2, #ta#:Center, #tbc#:#ffffff, #tc#:#000000, #tcv1#:70, #tcv2#:30, #tff#:Comic Sans MS, #th#:Auto, #titleShadow#:, #to#:1, #top1#:[1], #top2#:[3], #tp#:3,
#ts#:150, #tt#:Humidity, #ttr1#:?, #ttr2#:?, #tw#:90]
```

Overrides:

```
[#data#:transform: rotateX(10deg) rotateY(15deg);background: linear-gradient(45deg, #fff 0%, #000 50%,#fff 100%)]
```

Import

You can paste settings from other people in here and save them as a new sytle. How great is that!

Paste Basic Settings Here!

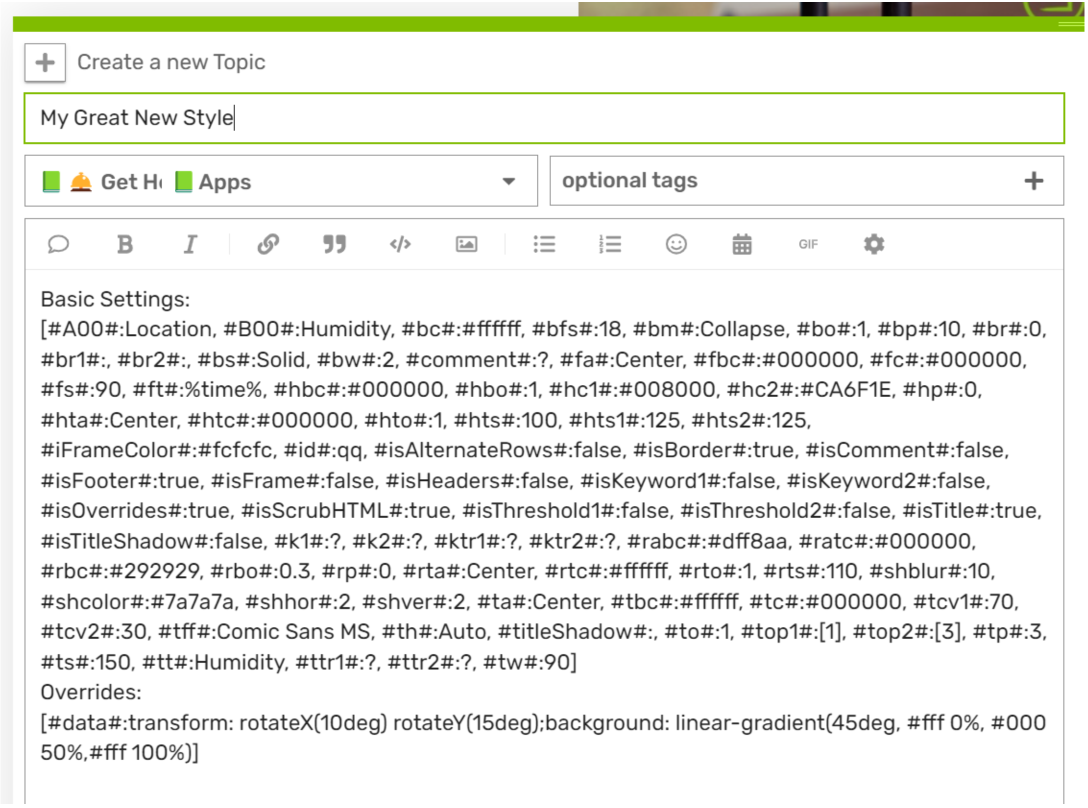
?

Paste Overrides Here!

?

Once you have imported a new Style you can save it if you wish to preserve it.

To share a style with others simply copy the text from **Basic Settings: down to the end of the Overrides section** and paste it into a forum message so it looks like this.



My Great New Style

Basic Settings:

```
[#A00#:Location, #B00#:Humidity, #bc#:#ffffff, #bfs#:18, #bm#:Collapse, #bo#:1, #bp#:10, #br#:0,
#br1#:, #br2#:, #bs#:Solid, #bw#:2, #comment#:?, #fa#:Center, #fbc#:#000000, #fc#:#000000,
#fs#:90, #ft#:%time%, #hbc#:#000000, #hbo#:1, #hc1#:#008000, #hc2#:#CA6F1E, #hp#:0,
#hta#:Center, #htc#:#000000, #hto#:1, #hts#:100, #hts1#:125, #hts2#:125,
#iFrameColor#:#fcfcfc, #id#:qq, #isAlternateRows#:false, #isBorder#:true, #isComment#:false,
#isFooter#:true, #isFrame#:false, #isHeaders#:false, #isKeyword1#:false, #isKeyword2#:false,
#isOverrides#:true, #isScrubHTML#:true, #isThreshold1#:false, #isThreshold2#:false, #isTitle#:true,
#isTitleShadow#:false, #k1#:?, #k2#:?, #ktr1#:?, #ktr2#:?, #rabc#:#dff8aa, #ratc#:#000000,
#rbc#:#292929, #rbo#:0.3, #rp#:0, #rta#:Center, #rtc#:#ffffff, #rto#:1, #rts#:110, #shblur#:10,
#shcolor#:#7a7a7a, #shhor#:2, #shver#:2, #ta#:Center, #tbc#:#ffffff, #tc#:#000000, #tcv1#:70,
#tcv2#:30, #tff#:Comic Sans MS, #th#:Auto, #titleShadow#:, #to#:1, #top1#:[1], #top2#:[3], #tp#:3,
#ts#:150, #tt#:Humidity, #ttr1#:?, #ttr2#:?, #tw#:90]
```

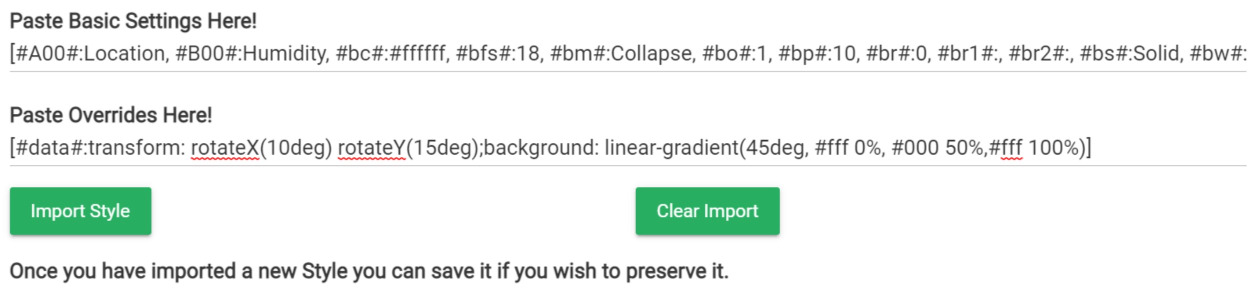
Overrides:

```
[#data#:transform: rotateX(10deg) rotateY(15deg);background: linear-gradient(45deg, #fff 0%, #000
50%,#fff 100%)]
```

That’s it, that is all you need to do to share a style with other people.

## Import Style

To import a style, you simply reverse the process described above. Copy the string values from a forum into the **Basic Settings** and **Overrides** fields and click **Import Style**.



Paste Basic Settings Here!

[#A00#:Location, #B00#:Humidity, #bc#:#ffffff, #bfs#:18, #bm#:Collapse, #bo#:1, #bp#:10, #br#:0, #br1#:, #br2#:, #bs#:Solid, #bw#:

Paste Overrides Here!

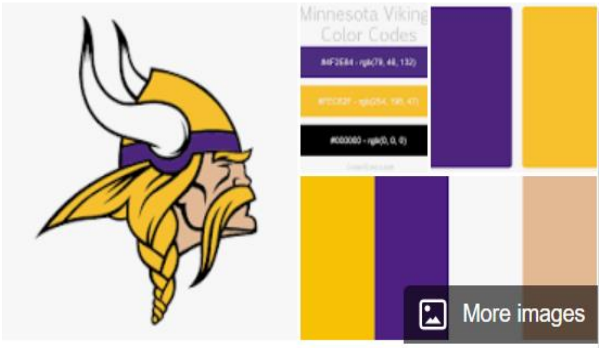
[#data#:transform: rotateX(10deg) rotateY(15deg);background: linear-gradient(45deg, #fff 0%, #000 50%,#fff 100%)]

Once you have imported a new Style you can save it if you wish to preserve it.

The new settings will be immediately applied. If you like the new style, you can save it to your style list for future use.

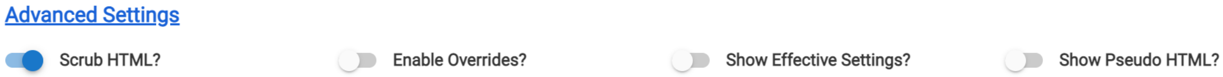
**A Note on Styles**

I'm not great at colors so the built-in styles reflect that. I'm hoping that others in the community with a better eye for color and design will share their creations for the benefit of the community and we can incorporate those as built in styles in future releases. What the built-in styles do offer is an example of what is possible with **Tile Builder** and hopefully spark your imagination to explore and make your own creations. It's quite a fun way to spend 15 minutes and you can use the eye-dropper tool to select colors from images on the internet such as these to create your personalized color scheme. These two swatches were used as the basis for the **Minnesota Vikings** and **Palm Springs** styles.



## Advanced\Overrides

The contents of the Advanced tab look like this.



Advanced Settings

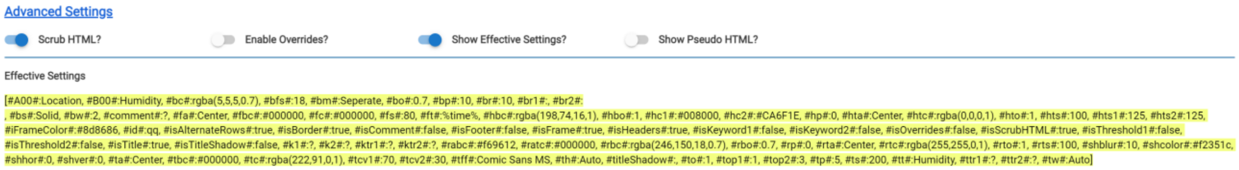
● Scrub HTML? ○ Enable Overrides? ○ Show Effective Settings? ○ Show Pseudo HTML?

### Scrub HTML

When this is enabled a scrubbing routine is employed to remove all excess characters from the HTML. For example, the default text alignment is left so any references explicitly setting this value can be removed. To see the default values for various HTML elements, load the style **Everything Off**, this will set all values to their default setting for the smallest table size.

### Show Effective Settings

This is a diagnostic tool that provides an easy way to view the effective settings which are a combination of the UI selected settings with the Overrides applied. Normally this can be left off.



Advanced Settings

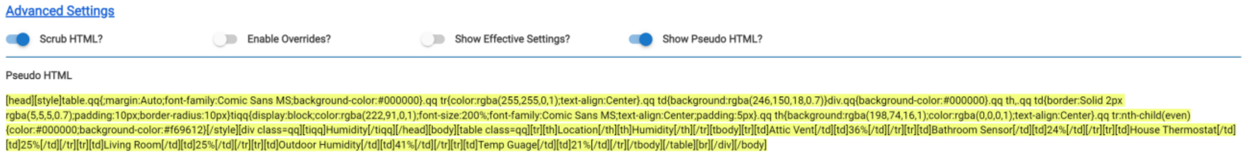
● Scrub HTML? ○ Enable Overrides? ● Show Effective Settings? ○ Show Pseudo HTML?

Effective Settings

```
[#A00#:Location, #B00#:Humidity, #bc#:rgba(5,5,5,0.7), #bfs#:18, #bm#:Seperate, #bo#:0.7, #bp#:10, #br#:10, #br1#:, #br2#:
, #bs#:Solid, #bw#:2, #comment#:?, #fa#:Center, #fbc#:#000000, #fc#:#000000, #fs#:80, #ft#:%time%, #hbc#:rgba(198,74,16,1), #hbo#:1, #hc1#:#008000, #hc2#:#CA6F1E, #hp#:0, #hta#:Center, #htc#:rgba(0,0,0,1), #hto#:1, #hts#:100, #hts1#:125, #hts2#:125,
#iFrameColor#:#8d8686, #id#:qq, #isAlternateRows#:true, #isBorder#:true, #isComment#:false, #isFooter#:false, #isFrame#:true, #isHeaders#:true, #isKeyword1#:false, #isKeyword2#:false, #isOverrides#:false, #isScrubHTML#:true, #isThreshold1#:false,
#isThreshold2#:false, #isTitle#:true, #isTitleShadow#:false, #k1#:?, #k2#:?, #ktr1#:?, #ktr2#:?, #rabc#:#f69612, #ratc#:#000000, #rbc#:rgba(246,150,18,0.7), #rbo#:0.7, #rp#:0, #rta#:Center, #rtc#:rgba(255,255,0,1), #rto#:1, #rts#:100, #shblur#:10, #shcolor#:#f2351c,
#shhor#:0, #shver#:0, #ta#:Center, #tbc#:#000000, #tc#:rgba(222,91,0,1), #tcv1#:70, #tcv2#:30, #tff#:Comic Sans MS, #th#:Auto, #titleShadow#:, #to#:1, #top1#:1, #top2#:3, #tp#:5, #ts#:200, #tt#:Humidity, #ttr1#:?, #ttr2#:?, #tw#:Auto]
```

### Show Pseudo HTML

This is also a diagnostic tool used for troubleshooting. When enabled the HTML is displayed in text form but using **[]** instead of **<>** so it can be viewed in text. You can copy this to an HTML editor and replace the **[]** with **<>** and it will render. Normally this can be left off.



Advanced Settings

● Scrub HTML? ○ Enable Overrides? ○ Show Effective Settings? ● Show Pseudo HTML?

Pseudo HTML

```
[head][style]table.qq{;margin:Auto;font-family:Comic Sans MS;background-color:#000000}.qq tr{color:rgba(255,255,0,1);text-align:Center}.qq td{background:rgba(246,150,18,0.7)}div.qq{background-color:#000000}.qq th,.qq td{border:Solid 2px
rgba(5,5,5,0.7);padding:10px;border-radius:10px}tiqq{display:block;color:rgba(222,91,0,1);font-size:200%;font-family:Comic Sans MS;text-align:Center;padding:5px}.qq th{background:rgba(198,74,16,1);color:rgba(0,0,0,1);text-align:Center}.qq tr:nth-child(even)
{color:#000000;background-color:#f69612}[/style][div class=qq][tiqq]Humidity[/tiqq][/head][body][table class=qq][tr][th]Location[/th][th]Humidity[/th][/tr][tbody][tr][td]Attic Vent[/td][td]36%[/td][/tr][tr][td]Bathroom Sensor[/td][td]24%[/td][/tr][tr][td]House Thermostat[/td]
[td]25%[/td][/tr][tr][td]Living Room[/td][td]25%[/td][/tr][tr][td]Outdoor Humidity[/td][td]41%[/td][/tr][tr][td]Temp Guage[/td][td]21%[/td][/tr][/tbody][/table][br][/div][/body]
```

## Overrides

Overrides are a way of replacing content within the HTML with something else. To better understand how overrides work consider this line of code from the **Tile Builder** application.

```
String HTMLTITLESTYLE = “<style>ti#id#{display:block;color:#tc#;font-size:#ts#%;font-family:#tff#;text-
align:#ta#;#titleShadow#;padding:#tp#px;#title#}</style>"
```

Each instance you see **#??#** is a field that will get replaced in the final HTML. For example **font-size:#ts#**. If we set the font size to 90% using the UI it will become **font-size:90%**. This occurs many times over for each of the different settings. There are many of these fields laid out in **Appendix A**.

### Simple Use Case

You won’t use **overrides** if the setting that you need is available within the menu system, but the menu system has a finite number of options so there are times when it makes sense.

- You can select row text to 70% or 75% but, what about the values in between? **#rts#=72**
- Use a font family not available in the **Tile Builder** UI. **#tff#=Blackadder ITC** (on Windows)

But these are edge cases and not super important.

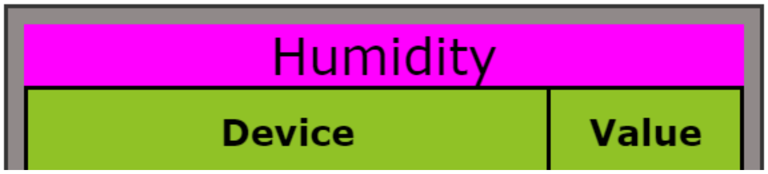
### Class Tags

Looking back at that line of code for the **Title block** you will also see an entry **#title#**. This is a placeholder that we can use to extend the properties of the title.

These entries require a property and a value, for example:

**#Title#= background-color: #ff00ff;**

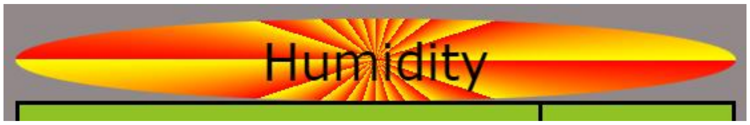
In this case **#title#** block will be replaced with **background-color: #ff00ff;** and the result looks like this.



Even though the **Title** field does not have a background property defined in the **Tile Builder** UI we can still add one using this method. We are not limited to a single command. In the next example we set a gradient behind the title and add a border radius to make an oval.

**#Title#=background-image: repeating-conic-gradient(red 10%, yellow 15%);border-radius: 55%**

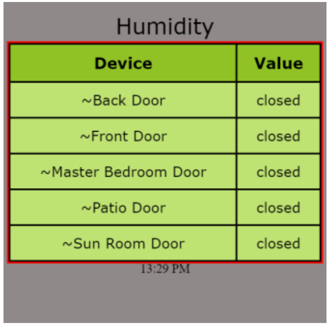
It produces this ugly result but you get the idea:



Each of the sections of the HTML table has its own class tag as follows. The **#???#=outline: 2px solid red;** is changed in each example to show which area is affected.

## Class Tag Examples

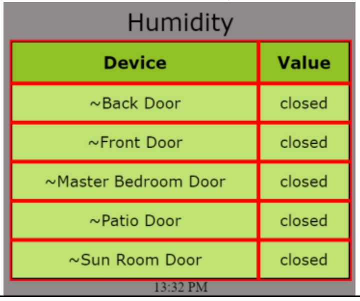
**#Table#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:29 PM

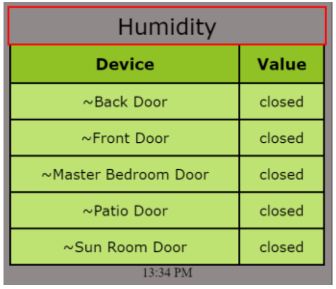
**#Border#=outline: 2px solid red;**
**Or #Row#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:32 PM

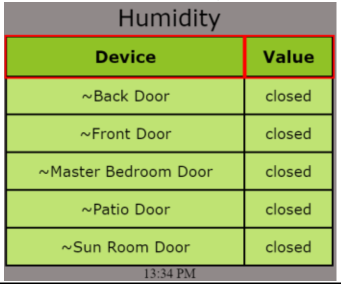
**#Title#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:34 PM

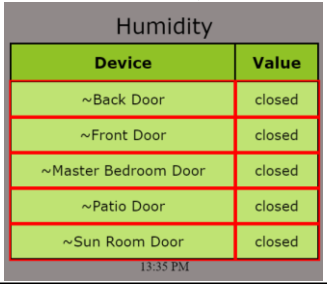
**#Header#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:34 PM

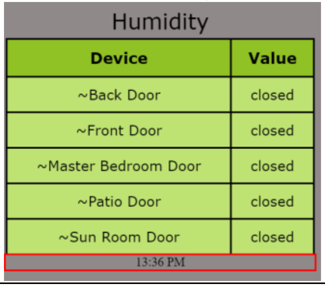
**#Data#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:35 PM

**#Footer#=outline: 2px solid red;**



| Humidity | |
|---|---|
| Device | Value |
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

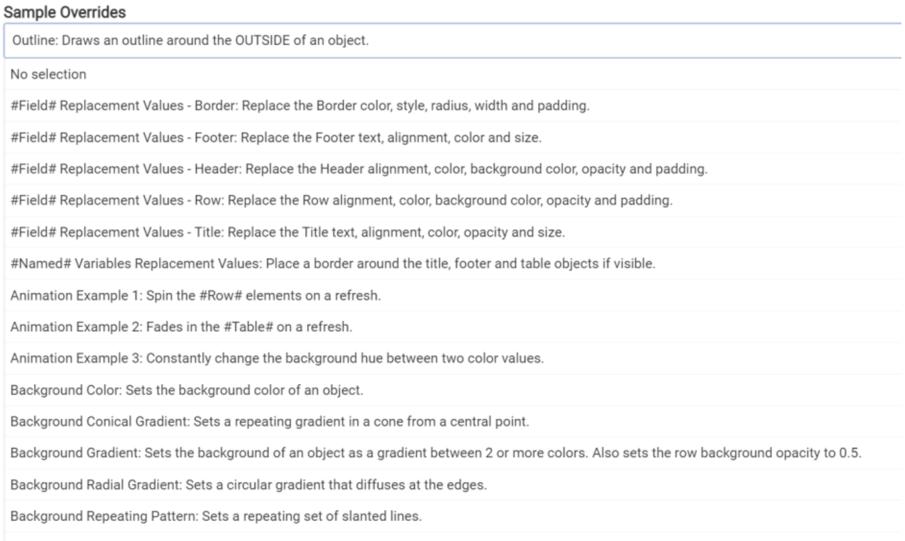
13:36 PM

The remaining tag **#Class#** is special purpose and will be covered later.

## Creating Effects

Where do these command strings come from? How do I know what I can put in here?

To get you started with **Overrides** a helper has about 40+ built in examples.



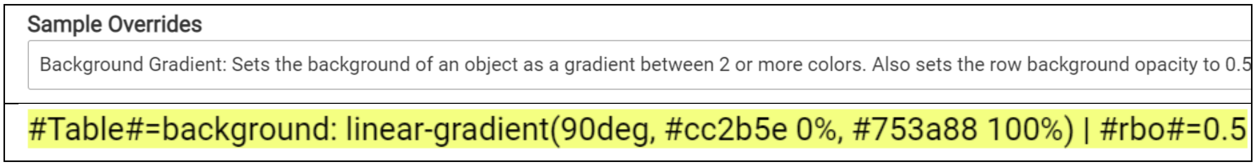
Sample Overrides

Outline: Draws an outline around the OUTSIDE of an object.

No selection
#Field# Replacement Values - Border: Replace the Border color, style, radius, width and padding.
#Field# Replacement Values - Footer: Replace the Footer text, alignment, color and size.
#Field# Replacement Values - Header: Replace the Header alignment, color, background color, opacity and padding.
#Field# Replacement Values - Row: Replace the Row alignment, color, background color, opacity and padding.
#Field# Replacement Values - Title: Replace the Title text, alignment, color, opacity and size.
#Named# Variables Replacement Values: Place a border around the title, footer and table objects if visible.
Animation Example 1: Spin the #Row# elements on a refresh.
Animation Example 2: Fades in the #Table# on a refresh.
Animation Example 3: Constantly change the background hue between two color values.
Background Color: Sets the background color of an object.
Background Conical Gradient: Sets a repeating gradient in a cone from a central point.
Background Gradient: Sets the background of an object as a gradient between 2 or more colors. Also sets the row background opacity to 0.5.
Background Radial Gradient: Sets a circular gradient that diffuses at the edges.
Background Repeating Pattern: Sets a repeating set of slanted lines.

Simply **select the Helper Sample** you wish to try, click on **Copy to Overrides** and then click **Refresh**. The displayed table will change to reflect the new settings. All the changes are created by the content of the overrides string applied to the present configuration.

Let us try this one which creates a gradient between two colors.

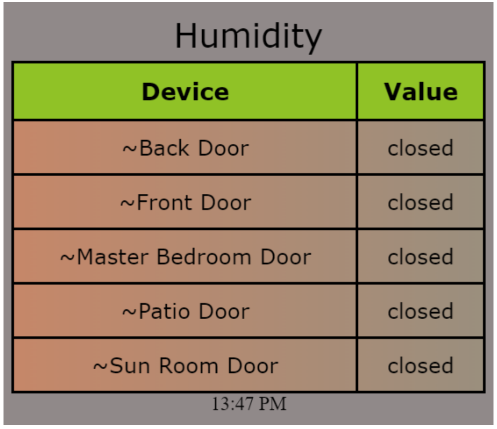
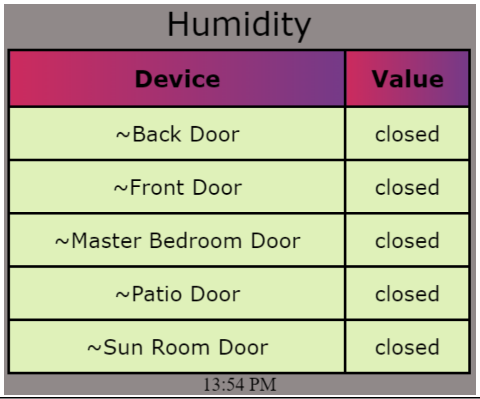


Sample Overrides

Background Gradient: Sets the background of an object as a gradient between 2 or more colors. Also sets the row background opacity to 0.5

```
#Table#=background: linear-gradient(90deg, #cc2b5e 0%, #753a88 100%) | #rbo#=0.5
```

**Copy to overrides and apply**.

| **Initial appearance with override applied.** | Edit the **Overrides** field and change **#Table#** to **#Header#**. |
|---|---|
|  |  |

Humidity

| Device | Value |
|---|---|
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

13:47 PM

Humidity

| Device | Value |
|---|---|
| ~Back Door | closed |
| ~Front Door | closed |
| ~Master Bedroom Door | closed |
| ~Patio Door | closed |
| ~Sun Room Door | closed |

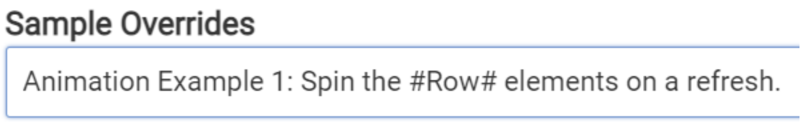
13:54 PM

Most of these examples I pieced together by using online CSS\HTML generators. One of the better ones I used can be found here: https://webcode.tools/generators/css . Looks complicated but it’s quite easy if you follow the example.

## Classes

Classes are small pieces of code that can be called upon to change an object. These are common in CSS and can be utilized in **Tile Builder Advanced**. Even if you have never programmed, I'll show you how easy it can be and the big effects they can have.

Let us take one of the sample overrides and break it down.



Sample Overrides

Animation Example 1: Spin the #Row# elements on a refresh.

```
#Class#=@keyframes myAnim {0% {opacity: 0;transform: rotate(-540deg) scale(0);}100% {opacity
1;transform: rotate(0) scale(1);}} | #Row#=animation: myAnim 2s ease 0s 1 normal forwards;
```

**Class Sample Part 1**

The first part is the class and what is shown in red will be substituted into the HTML code in place of the **#Class#** placeholder.

> **#Class#**=@keyframes myAnim {0% {opacity: 0;transform: rotate(-540deg) scale(0);}100% {opacity: 1;transform: rotate(0) scale(1);}}

**@keyframes** indicates that it going to be a transition between 2 or more points enclosed in {}

**myAnim** is the name of the animation class and is used to activate it.

The first point is: **0% {opacity: 0;transform: rotate(-540deg) scale(0)}**

- 0% marks it as the starting point of the transformation.
- Transform means it's a transformation of the object with the following properties.
- Rotate (-540deg) means it will begin pre-rotated 540 degrees to the left.
- Scale (0) means that it will not change in size at its starting point.

The second point is: **100% {opacity: 1;transform: rotate(0) scale(1)}**

- 100% marks it as the ending point of the transformation.
- Transform means it's a transformation of the object with the following properties.
- Rotate (0deg) means it will end in a normal position (0 degrees rotated).
- Scale (0) means that it will not change in size at its ending point.

This animation will spin the object from -540 degrees to 0 degrees when implemented. The browser will make the animation smooth between the starting and ending points. It is possible to add other points in the animation, at say 50% with different properties if so desired, but they take up more precious space.

**Class Sample Part 2**

The '**|**' is just a required separator that **Tile Builder** uses to handle multiple instruction on a single line. It will be stripped out and discarded.

**Class Sample Part 3**

The last part is used to associate the class with some part of the HTML table, in this case the **#Row#** element.

**#Row#=animation: myAnim 2s ease 0s 1 normal forwards;**

This will cause **Tile Builder** to look for the **#Row#** placeholder and replace it with **animation: myAnim 2s ease 0s 1 normal forwards;**

**animation**: is a required element for HTML\CSS indicating the presence of the animation.

**myAnim**: Is a name and it must match the name of a class, in this case the name given earlier.

**2s ease 0s 1 normal forwards;**: These are parameters passed to the animation that control things like animation speed, duration, direction, reversal nature etc.

That is how to implement and call a class. The class will get activated at refresh.

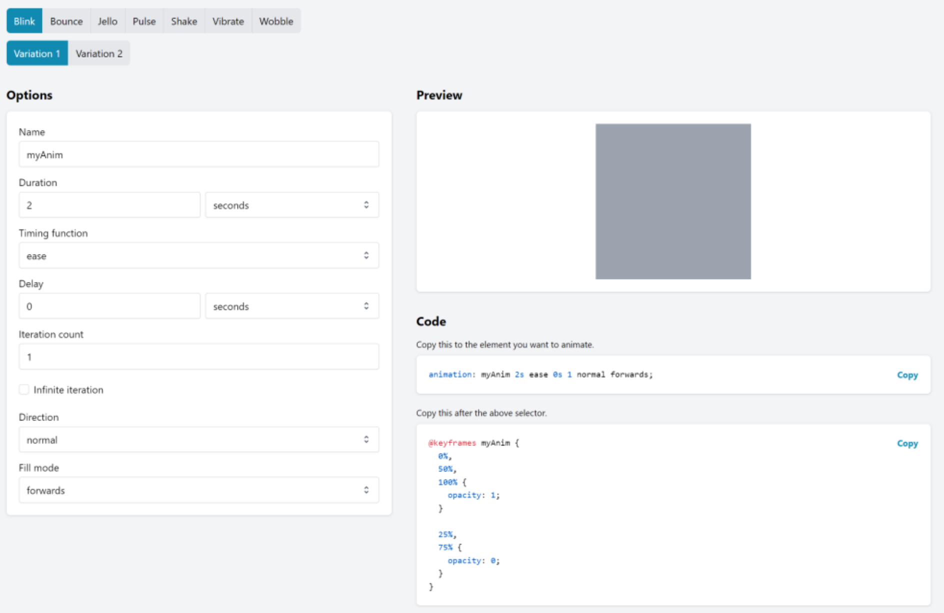
**Very Important:** The class and animation names must be unique across all tiles unless they are intentionally re-used. In the above examples instead of using **myAnim** you might use **spin** or **rotate**.

## Online Resources for Classes

You don't need to write these commands from scratch. You can use a generator like this.

https://webcode.tools/generators/css/keyframe-animation

Enter the parameters that you want, and it will generate the command strings as shown below.



**Options**

Name: myAnim

Duration: 2 seconds

Timing function: ease

Delay: 0 seconds

Iteration count: 1

□ Infinite iteration

Direction: normal

Fill mode: forwards

**Code**

Copy this to the element you want to animate.

```
animation: myAnim 2s ease 0s 1 normal forwards;
```

Copy this after the above selector.

```
@keyframes myAnim {
  0%,
  50%,
  100% {
    opacity: 1;
  }

  25%,
  75% {
    opacity: 0;
  }
}
```

Turn the above strings into:

**#Class#=@keyframes myAnim {0%,50%,100% {opacity: 1;}25%,75% {opacity: 0;}}**

**#Title#=animation: myAnim 2s ease 0s 1 normal forwards;**

Put them together with the '|' separator and paste the string into the **overrides** field like this.

```
#Class#=@keyframes myAnim {0%,50%,100% {opacity: 1;}25%,75% {opacity: 0;}} |
#Title#=animation: myAnim 2s ease 0s 1 normal forwards;
```

You now have a title that blinks twice whenever the table is refreshed. This happens automatically on the dashboard when the table gets new data.

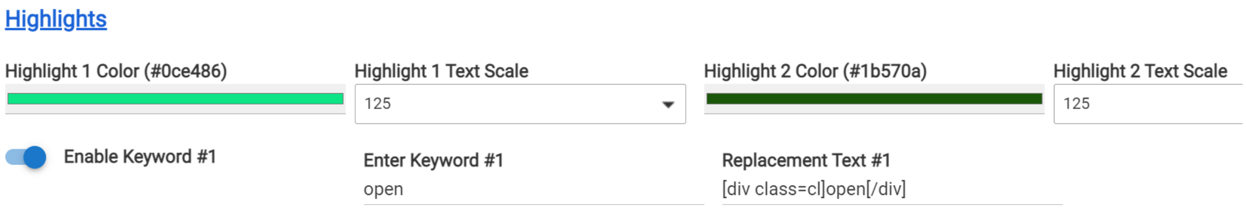
**Important: When generating an event to test an animation it will take effect immediately on the dashboard due to the event subscriptions. This is not true within the Tile Builder UI so you must do a refresh manually to see event changes.**

## Static Class Example

Classes can do all kinds of things and are not just limited to animation. In this example we create a class called "**cl**" which turns the background color to red. Note we use a leading period so in the declaration it's "**.cl**" but referred to by using "**cl**". This name can be anything, but shorter names save precious space especially when repeated across multiple rows.

**#Class#=.cl{background-color: #ff0000;}**

We can call this class when we get a matching condition in **Highlights** by doing the following. Whenever the word **open** is detected, we call the class to change the background color.



Highlights

Highlight 1 Color (#0ce486)

Highlight 1 Text Scale: 125

Highlight 2 Color (#1b570a)

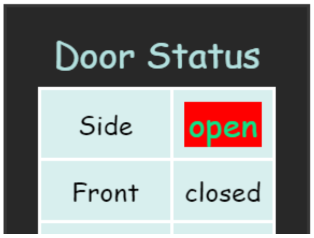
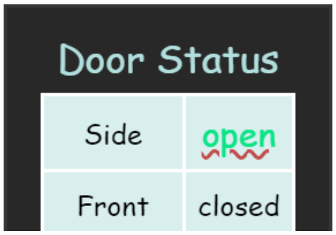
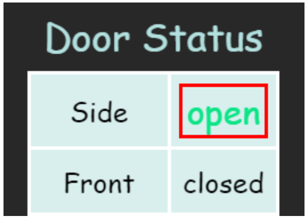
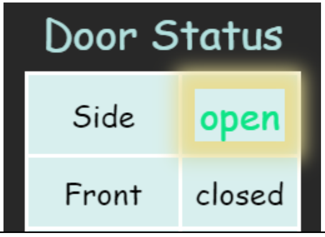
Highlight 2 Text Scale: 125

Enable Keyword #1

Enter Keyword #1: open

Replacement Text #1: [div class=cl]open[/div]

The above example result is shown in the first cell. The required class definition is shown for other effects. The results looks like this:

| | |
|---|---|
| **#Class#=.cl{background-color: #ff0000;}**<br><br>Door Status<br>Side open<br>Front closed | **#Class#=.cl{text-decoration: underline wavy #C34E4E;}**<br><br>Door Status<br>Side open<br>Front closed |
| **#Class#=.cl{outline: 2px solid red;}**<br><br>Door Status<br>Side open<br>Front closed | **#Class#=.cl{box-shadow: 0px 0px 10px 10px #E8DD95;}**<br><br>Door Status<br>Side open<br>Front closed |

If we were to look inside the HTML we would see the code **[div class=cl]open[/div]** repeated on each line where a contact is open. Naturally this takes up more space than just **open**. When you are considering these enhancements, it is best to apply these effects to a small number of results where space is a concern.

## Animated Icon Examples

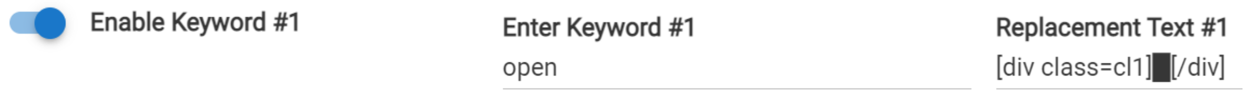
The following line of code is an animation that will move an element 20 pixels to the left at the start and end at 20 pixels to the right of its normal position.

> ***animation: myAnim 1s linear 0s infinite alternate-reverse;} @keyframes myAnim {0% {transform: translateX(-20px);}100% {transform: translateX(20px);}***

We can use it as a class by doing this in overrides. The class name is cl1.

> **#Class#=.cl1{animation: myAnim 1s linear 0s infinite alternate-reverse;} @keyframes myAnim {0% {transform: translateX(-20px);}100% {transform: translateX(20px);}}**

Using the same trick as the prior example we can add this animation for any open door.

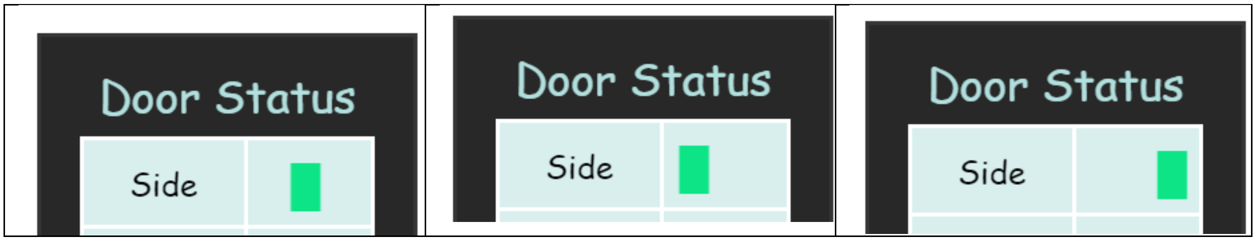


Enable Keyword #1

Enter Keyword #1: open

Replacement Text #1: [div class=cl1]█[/div]

Using an ascii block character combined with the animation class we get a pleasing blob that moves back and forth smoothly and continuously until the door is closed.



You don't have to write these statements by hand. Use the **Overrides Helper** examples or an HTML\CSS generator such as [https://webcode.tools/generators/css/keyframe-animation](https://webcode.tools/generators/css/keyframe-animation).

## Spinner Example

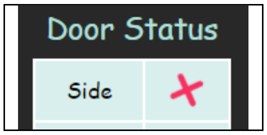
Spinners are commonly used to indicate activity and a variety of characters are suitable. I chose the emoji character ❌ but extended ascii characters such as ☼ , 🌀 , ✳ are well suited to indicating a spinning motion. With Unicode characters you also get the option of specifying the character color and size. With emojis you can only specify size. Try this out.

Copy this line to your overrides field and press enter.

> **#Class#=.cl1{animation: myAnim 1s linear 0s infinite normal forwards;} @keyframes myAnim {0% {transform: rotate(0deg);}100% {transform: rotate(360deg);}}**

Change the replacement text in **Highlights** to **[div class=cl1]❌[/div]**

The result is a red X that will spin continuously when the door is open. Spinners are a good option for motion, contacts, switches, fans etc.



## A Word About Classes

Before you start creating classes there are a couple of things you must know.

### Scope

The scope of a class is global. If you set up a class via tile A you can reference it using tile B. The good news is that can save space, the bad news is it can have unintended consequences if you are not careful about naming your classes.

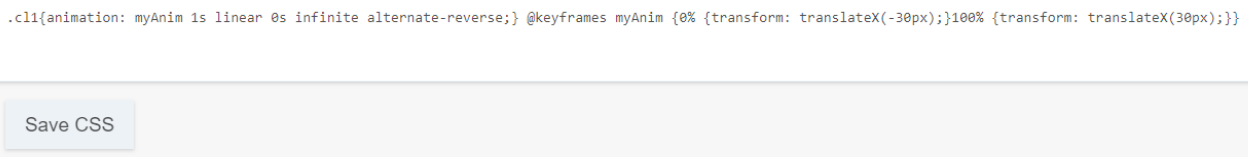
### Naming

Each unique class or animation must have a unique name. Because of space constraints I'd recommend you name classes cl1, cl2, cl3 etc. and animations an1, an2, an3… etc. This should avoid naming collisions. If you have two classes by the same name anywhere in the dashboard only one of them will work.

### Offloading Classes

I've tried to make **Tile Builder** accessible to all **Hubitat** users and have avoided any mention of directly editing the Hubitat CSS, but I must acknowledge that capability exists and could be very helpful in standardizing classes across multiple tiles or shrinking the tile size by shifting the burden of the classes from a tile to the dashboard CSS code and keeping everything in one place.

The screenshot below shows a class called "**cl1**" added to the Hubitat Dashboard CSS.



```
.cl1{animation: myAnim 1s linear 0s infinite alternate-reverse;} @keyframes myAnim {0% {transform: translateX(-30px);}100% {transform: translateX(30px);}}
```

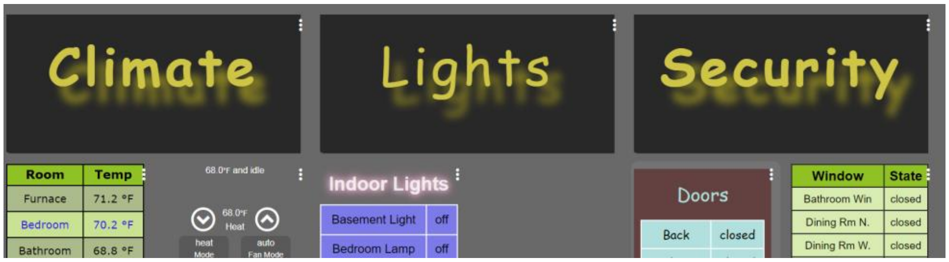
The above class was used earlier to indicate motion. It is implemented by going to **Highlights** and specifying the **Keyword** of "**active**" be replaced by the specified text **[div class=cl1]█[/div]**. In this case the █ **character** would move back and forth when the motion sensor is in an active state.

## Advanced Techniques

This section covers some techniques I have discovered. Even though I wrote it I'm still finding new ways of doing things which may be of interest to others.

### Create a Menu Block

I like to group my topics on the dashboard. In the picture below I've chosen to do vertical groupings and put a title block at the top.



| Room | Temp |
|---|---|
| Furnace | 71.2 °F |
| Bedroom | 70.2 °F |
| Bathroom | 68.8 °F |

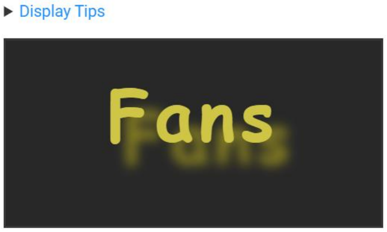
| Basement Light | off |
|---|---|
| Bedroom Lamp | off |

| Back | closed |
|---|---|

| Window | State |
|---|---|
| Bathroom Win | closed |
| Dining Rm N. | closed |
| Dining Rm W. | closed |

Here's how to do this.

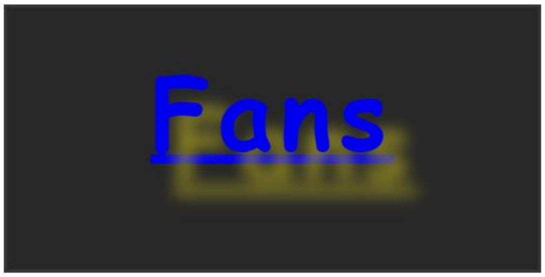
1. Add a new **Attribute Monitor** but don't select any attributes or devices.
2. Set the **Device Limit Threshold** to 0
3. Go to **Styles** and select the style **Menu Bar** and apply it**.**
4. Got to the **Title** tab and change your text. I chose fans and it looks like this.

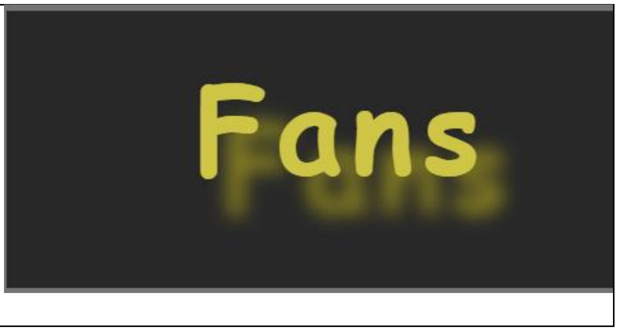


5. Give the tile a name and select a tile number to publish it on and then add it to your dashboard.
6. If you want to use it as a dashboard link you can by doing this.
   a. Edit the tile and go to the Title tab.
   b. Get the link to the dashboard and enter it like this:

   *[a href='http://192.168.0.200/apps/api/3204/dashboard/3204?access_token=1d3a0f0b-0caa-4bba-9cbb-c2fe44a4991b&local=true']Fans[/a]*

Your tile will now look like this in the preview window and on the dashboard.





## It’s a Wrap

Well, if you made it this far you are ready to exploit most of the power of **Tile Builder** to build beautiful, functional, and animated tiles to make your **Hubitat** dashboard a lot more fun and engaging. I look forward to seeing some of the designs that people come up with and share on the community forums.

I have several ideas for other **Tile Builder** modules of the same quality as **Activity Monitor** and **Attribute Builder**. I will almost certainly write them for my own use, but their public release will depend on the **Hubitat** communities’ willingness to acknowledge value in quality software and donate towards the ongoing development of the project.

This is the day the Lord has made, let us rejoice and be glad.

# Appendix A

# Tile Builder Field Code Reference

| General | Code | Units | Example | Notes |
|---|---|---|---|---|
| Table Width | #tw# | % | #tw#=85 | Auto or a % of the container width. |
| Table Height | #th# | % | #th#=90 | Auto or a % of the container height. |
| Border Mode | #bm# | String | #bm#=Separate | Options are Collapse and Separate |
| Background Color | #tbc# | #Hex | #tbc#=#e88c8c | This background color is visible when border mode is separate. |
| Font Family | #tff# | String | #tff#=Arial | Text Font Family for all text |

| Header | Code | Units | Example | Notes |
|---|---|---|---|---|
| Text | #A0# for Column 1 & #B0# for Column 2 | | #A0#=My Title | Any text string. You can include Emoji or HTML codes using square brackets [b]Bold[/b]. |
| Text Size | #hts# | % | #hts#=125 | Any integer value. |
| Text Color | #htc# | #Hex | #htc#=#000000 | Any Hex color. |
| Background Color | #hbc# | #Hex | #hbc#=#232323 | Any Hex color |
| Text Alignment | #hta# | String | #hta#=Left | Options are: Left, Center, Right and Justify |
| Text Opacity | #hto# | Float | #hto#=0.7 | An opacity value in the range 0 – 1. These are combined with the Hex colors to form RGBA values in the HTML. |
| Padding | #hp# | Px | #htp#=10 | An integer, typically in the range of 0 – 10. N/A if border padding is active. |

| Footer | Code | Units | Example | Notes |
|---|---|---|---|---|
| Text | #ft# | String | #ft#=My Footer | The footer string. You can use %day%, %time% as a macro. |
| Size | #fs# | % | #fs#=75 | Percentage of the base font size. |
| Color | #fc# | #Hex | #fc#=#5c290d | The color of the footer text |
| Alignment | #fa# | String | #fa#=Right | The alignment of the footer text. |
| | | | | |

| **Title** | **Code** | **Units** | **Example** | **Notes** |
|---|---|---|---|---|
| Text | #tt# | String | #tt#=My Title | The Title Text |
| Size | #ts# | % | #ts#=150 | Expressed as a % of the base font size. |
| Color | #tc# | #Hex | #tc#=#00FF00 | Color expressed as a 6 digit Hexadecimal |
| Alignment | #ta# | String | #ta#=Right | Options are: Left, Center, Right and Justify |
| Padding | #tp# | Integer | #tp#=3 | Change the space around Title. |

| **Border** | **Code** | **Units** | **Example** | **Notes** |
|---|---|---|---|---|
| Style | #bs# | String | #bs#=Solid | The border style |
| Width | #bw# | Pixels | #bw#=5 | The border width in pixels |
| Color | #bc# | #Hex | #bc#=#00FF00 | The border color |
| Radius | #br# | Integer | #br#=20 | The border radius in pixels |
| Padding | #bp# | Integer | #bp#=3 | The border padding in pixels |

| **Rows** | **Code** | **Units** | **Example** | **Notes** |
|---|---|---|---|---|
| Text Size | #rts# | % | #rts#=100 | Expressed as a % of the base font size. |
| Text Color | #rtc# | #Hex | #rtc#=#000000 | Any Hex color. |
| Text Alignment | #rta# | String | #rta#=Center | Options are: Left, Center, Right and Justify |
| Text Opacity | #rto# | Float | #rto#=0.8 | An opacity value in the range 0 – 1. These are combined with the Hex colors to form RGBA values in the HTML. |
| Padding | #rp# | Px | #rtp#=5 | An integer, typically in the range of 2 – 10. Not applicable if border padding is active. |
| Background Color | #rbc# | #Hex | #rbc#=#FF00FF | Background color of the row area. |
| Background Opacity | #rbo# | Float | #rbo#=0.5 | Background Opacity. |
| Alternate Text Color | #ratc# | #Hex | #ratc=#0000FF | The alternate text color when alternate table rows are configured. |
| Alternate Background Color | #rabc# | #Hex | #rabc#=#D3D3D3 | The alternate background color when alternate rows are configured. Change background opacity to make visible. |