Tile Builder Rooms Help
Version 1.0.0

Revised 8/31/23

Table of Contents

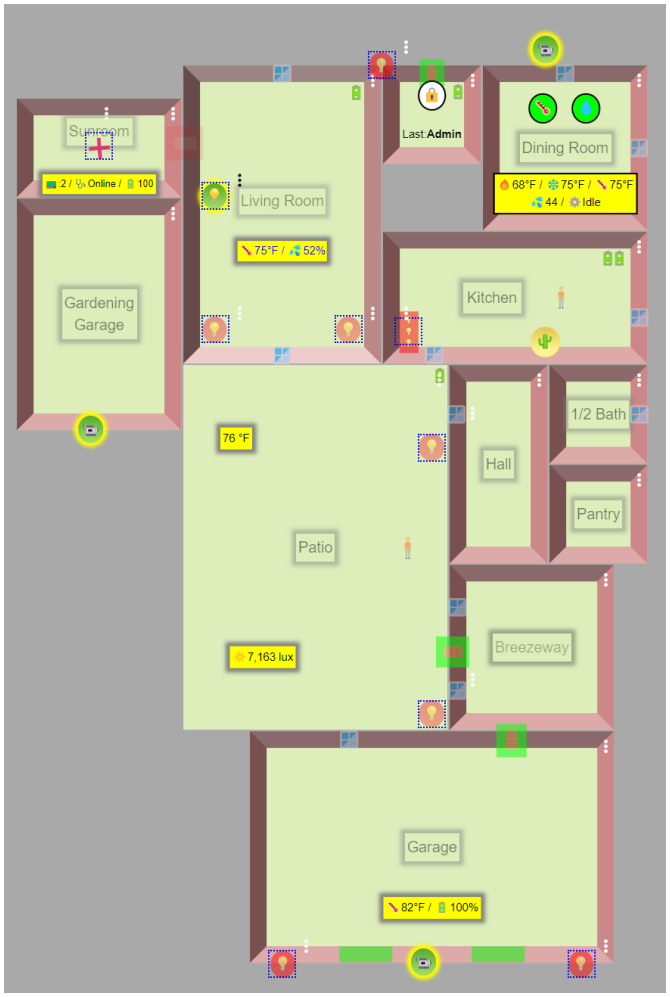
Section 1

Tile Builder Rooms

Introduction and Installation

Introduction

Rooms is a Tile Builder child app that allows users to create “Rooms” in which their Hubitat devices are presented in a geographic layout. The Icons representing devices change in multiple user configurable ways to indicate the present state of a device. Multiple rooms can be connected to generate a complete house floor plan as shown below.



Tile Builder Rooms is entirely native to Hubitat® and does not require any third-party software, hardware or even any knowledge of CSS to achieve results like those shown above.

Main Benefits of Tile Builder Rooms

- A native solution for Hubitat® dashboards that is easy to use and looks great.
- Full control over Icons, color, style, placement, effects etc.
- Allows a geographic layout of devices.
- Allows most Hubitat device data to be made visible.
- Tiles remain below the 1,024 byte limit so Tile Builder rooms work fully within the Hubitat app whether local or over the internet without a VPN.
- Entirely local to the hub means it’s fast and secure. No Maker API needed.
- Layer Hubitat devices over the Rooms if you want to add device control.

How Tile Builder Rooms Works

There are three components required for **Tile Builder Rooms**.

1) **Tile Builder Parent App** – The organizing parent app.
2) **Tile Builder Storage Driver** – Device driver used for storing **Tile Builder Tile** data.
3) **Tile Builder Child App – Rooms** (child app) – Generates the rooms for publishing to a dashboard.

The **Tile Builder** parent app is the primary organizing app under which all others are created.

Tile Builder - Rooms	Tile Builder (user)
7 Digit Font Test - Tile 1	Tile Builder - Multi Attribute Mon
Auto Fan Status - Tile 2	Tile Builder - Multi Attribute Mon
Batteries	Tile Builder - Attribute Monitor (u
Breezeway - Tile 18	Tile Builder - Rooms (user)
Dining Room - Tile 24	Tile Builder - Rooms (user)
Downstairs Bathroom - Tile 20	Tile Builder - Rooms (user)
Front Entry - Tile 23	Tile Builder - Rooms (user)
Garage - Tile 17	Tile Builder - Rooms (user)
Gardening Garage - Tile 15	Tile Builder - Rooms (user)
Hall - Tile 21	Tile Builder - Rooms (user)

Tiles are generated by one of the 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.

The Hubitat® dashboard has a limit of 1,024 bytes for any **attributes** that are displayed. The **Tile Builder Rooms** module enforces this limit and does not allow publication of tiles that exceed 1,024 bytes. Tile Builder Rooms generates extremely compact HTML and 1,024 bytes is enough for most rooms. If you have a room that would exceed this size limit, see the advanced techniques section for how to work around this.

Whenever you build a tile, the size of the tile and which components are active is displayed along the bottom.

Current HTML size is: **930** bytes. Maximum size for dashboard tiles is **1,024** bytes.

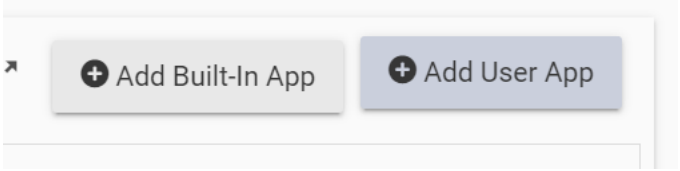
Enabled Features: Title: **On**, Walls: **On**, Room Device Count:**7**, IconBar A: **On** (**2**), IconBar B: **Off** (**0**)

Tile Builder Installation

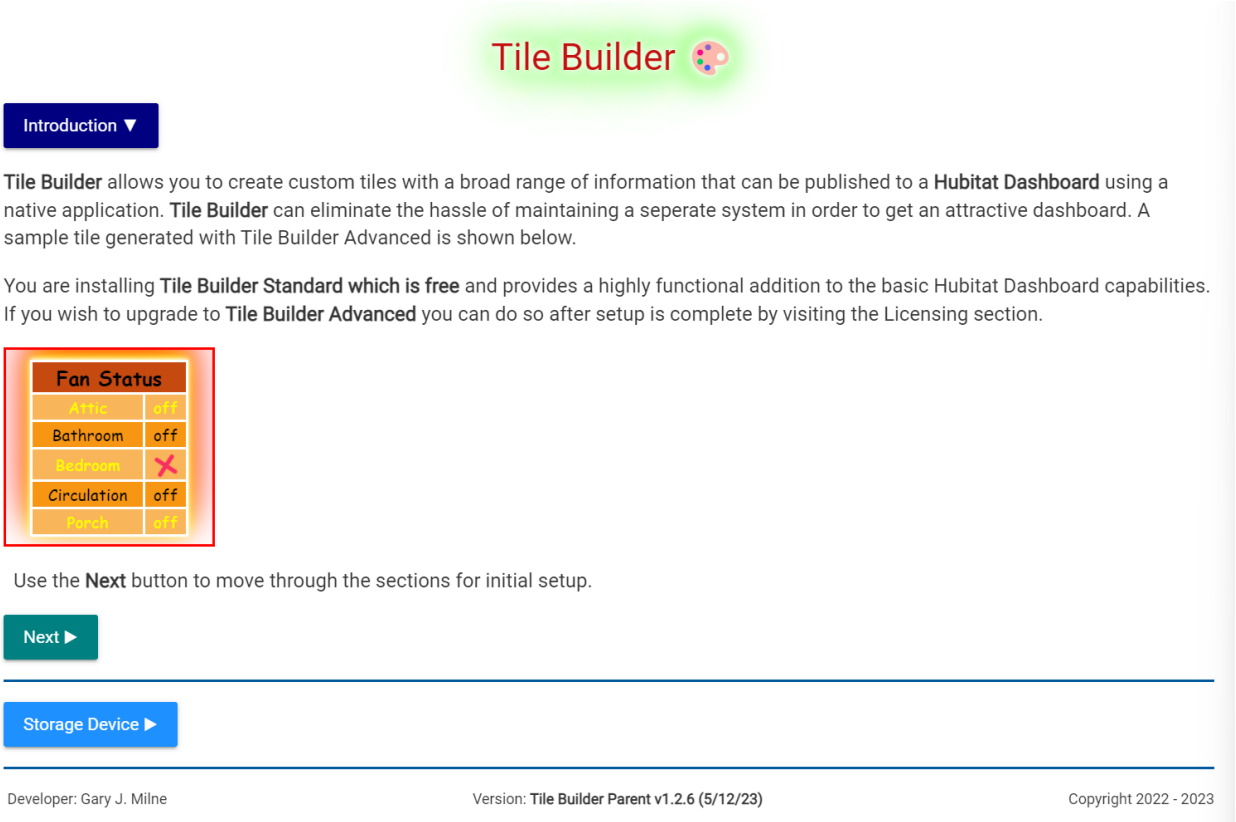
Tile Builder is listed in Hubitat® Package manager. Choose to install by tags and select the **Dashboards** tag. Select **Tile Builder for Hubitat®** and complete the installation process. This will place the code on your hub and then there are a few steps to complete the installation.

> 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**

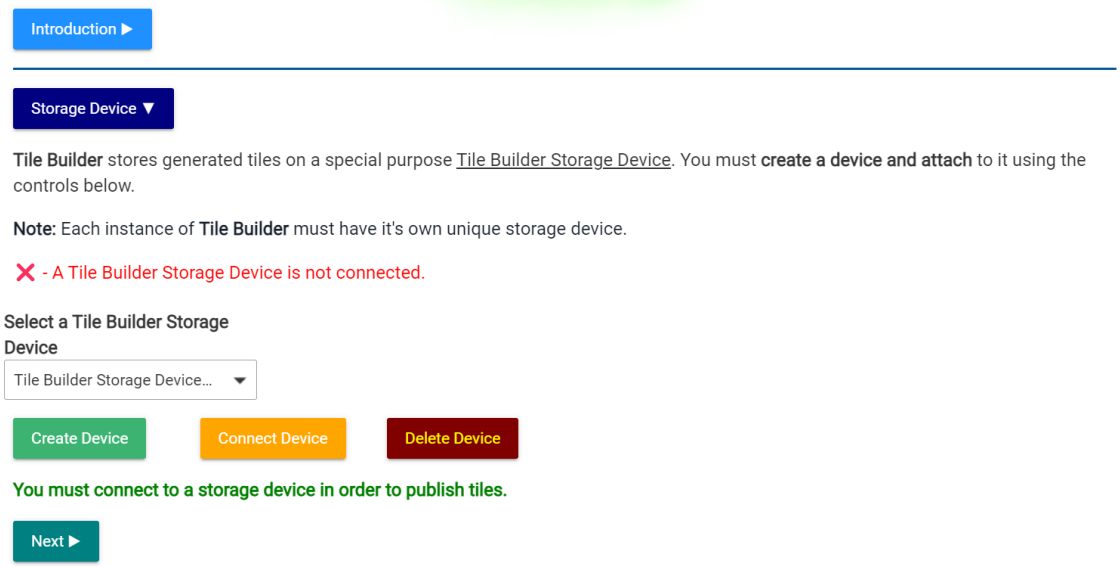


2. Select **Tile Builder** from the list of available apps.
3. All **Tile Builder** child modules are optional so be sure to select Rooms as an option.
4. **Tile Builder** will install and bring you to the parent screen.

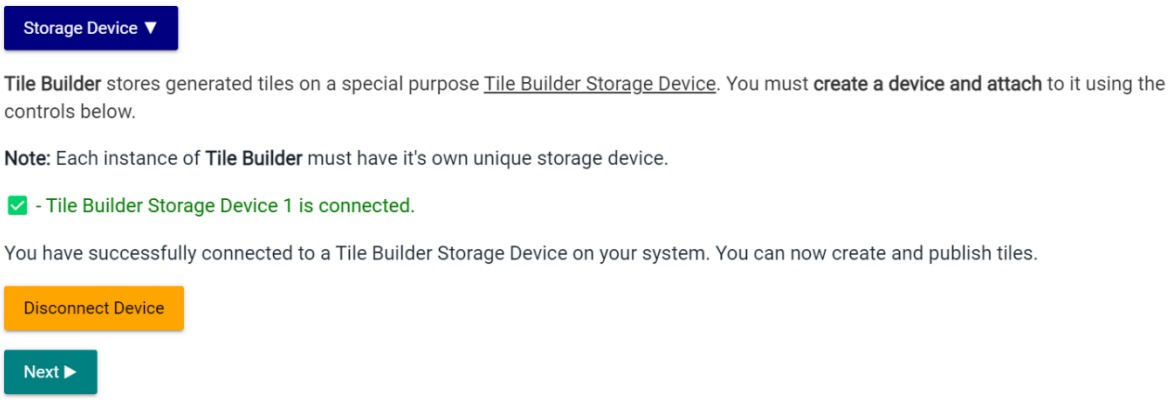


Click Next.

5. Create and Connect a Storage Device

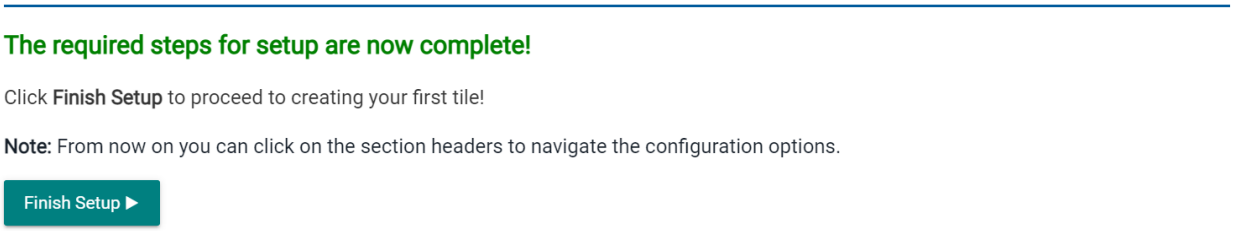


Once the device is created and connected it will look like this.



Click Next.

6. Finish Setup

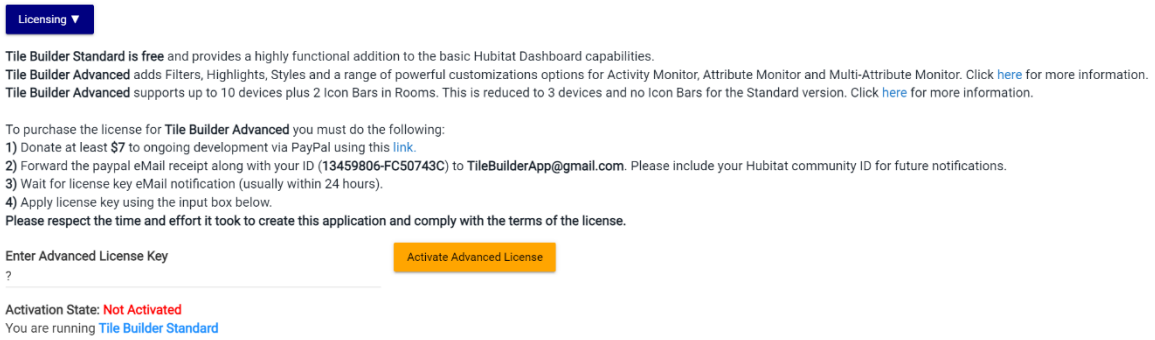


We can now create our first Room\Tile.

Licensing

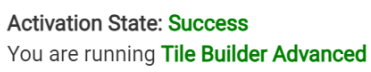
Tile Builder has **Standard** and **Advanced** versions. In the other **Tile Builder** modules, the Standard version has a pretty high degree of functionality. In **Rooms** the **Standard** version is limited to 3 devices with the intention of it being used primarily to build simple animated tiles. The **Advanced** can have up to 10 devices plus 2 Icon Bars which each support up to 5 devices.

If you only have the Standard (free) license for Tile Builder, you can still create very interesting and useful Tiles for your dashboard. The general principles are the same but some more specific examples are listed in the section called **Tile Builder Rooms – Standard**.



A lifetime hub license for **Tile Builder Advanced** is currently $7. Just follow the on-screen instructions.

Once activated the screen changes to look like this:



If you don't want to pay $7 you are still free to use **Tile Builder Standard** as much as you wish.

Pricing

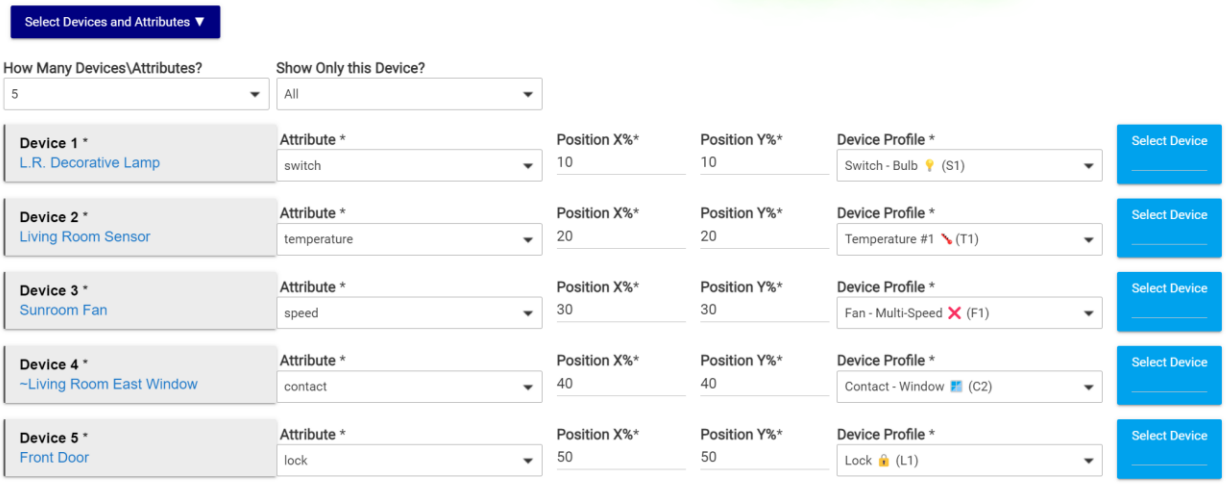
Tile Builder Advanced was initially priced at $5 when it launched in May 2023. The price was increased to $6 when Multi Attribute Monitor was released in July and is now increasing to $7 with the release of Rooms in September 2023. The price will continue to increase by $1 for each new module that is added, and I currently have plans for 2 more. Once you have an advanced license all future modules are included for as long as you own your hub.

Section 2

Tile Builder Rooms

Creating a Room

Within the **Tile Builder** parent app go to the section called **Create\Edit Tile** and select **Add New Room**. After about 15 seconds the **Rooms** main screen will be displayed. In the picture below I have chosen to have 5 devices and can now select the devices and attributes I'm interested in such as those shown below.



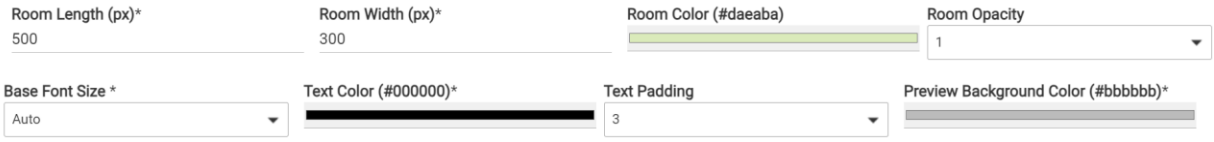
For each line you must select a device, attribute, X position, Y position and a device profile. These are explained below.

- **Device**: Any Hubitat device.
- **Attribute:** A list of attributes that the selected device supports.
- **Position X%:** The position of the Icon on the X axis normally expressed as a percentage between the far left (0%) and far right (100%). Position also supports decimal points and negative values.
- **Position Y%:** The position of the Icon on the Y axis normally expressed as a percentage between the top (0%) and bottom (100%). Position also supports decimal points and negative values. Thus, a position of 0,0 is the top left-hand corner and 50,50 is the center etc.,
- **Device Profile:** A device profile is a collection of properties to represent a device in each state. For example, a switch has two states, off or on. The device profile will define properties for the appearance of the device in each of those two states. A red background might indicate a switch is off and a green background might indicate a switch is on. These settings are fully customizable. These will be covered in more detail later, for now we are going to accept the default settings.
- **Select Device:** This button is a shortcut way of focusing the screen on just the selected device and the selected profile properties (not displayed above) to make better use of screen space.
- **Clear this Device:** Deletes the last device from the list. (Not shown)
- **Show All Devices:** Reverses the filter applied by the **Select Device** action. (Not shown)

Room Defaults

General Tab

Here we can adjust the proportions of the Room along with color, size etc. It is an important concept to note that once a room is placed on the Dashboard it will fill the container it is placed in. So strictly speaking the Room dimensions given here only affect the preview window and the final size will be dictated once it is placed on the Dashboard.



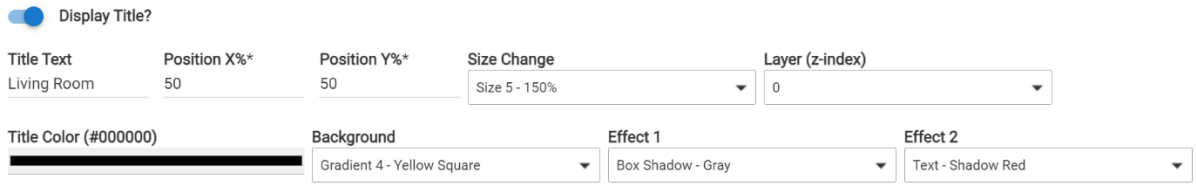
I will comment on a couple of these:

Room Opacity: You can make the “floor” of the room partially or fully transparent. By doing this you can place informational items “behind” the room, such as a Tile Builder Table, to make the dashboard more compact and useful.

Base Font Size: All the icon\text sizes in TB Rooms are expressed in percentages relative to this base measure. Typically the only time this should be changed is if you adjust the default “Font Size” on your dashboard to a different value and you need to adjust the preview size of the text to match.

Title Tab

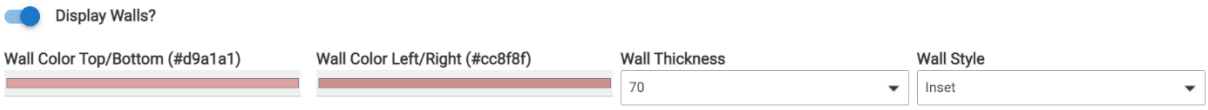
Here we can choose to display a title within the Room, probably a room name.



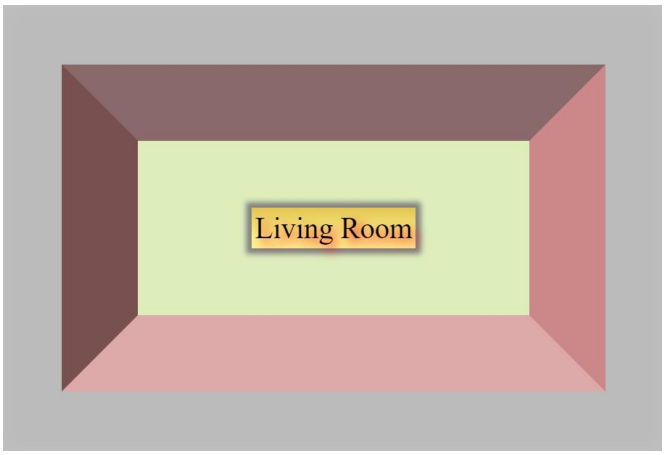
We can create an attractive Title to display within the room with just a few mouse clicks.

Walls Tab

Here we can enable the walls and modify their color, thickness, and style.

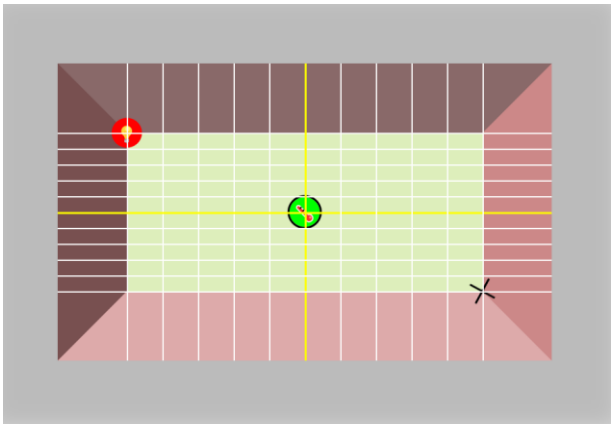


By increasing the thickness of the Walls you can enhance the illusion of looking down into a room.



Positioning Items

Items are positioned in the room using X and Y coordinates with 0%,0% being the upper left corner and 100,100 being the lower right corner. Icons are placed onto the grid using the exact center of an icon. So, an icon placed at 50%,50% would have ¼ of the icon in each quadrant. In the image below icons are placed at 0%,0% ; 50%,50% ; and 100%,100% to demonstrate this principle.

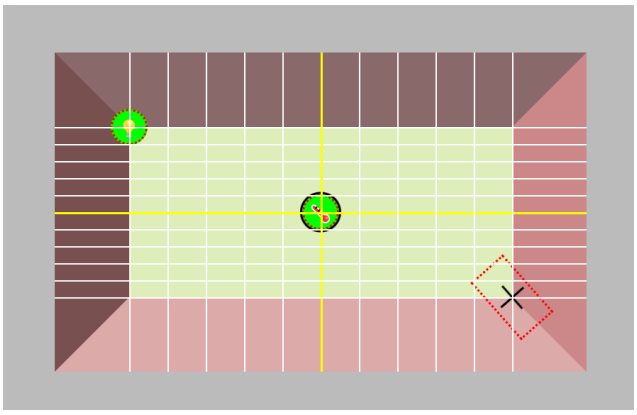


Positioning Grid

In the above graphic the positioning grid has been turned on as a visual aid. Notice the 0,0 – 100,100 grid area only covers the “floor” of the room. To position past the edges of the grid you must use coordinates outside the 0-100 range such 20%, 110% or -5%, 50%.

Show Object Boundaries

Another positioning aid is the ability to show object boundaries. These can help to explain issues with alignment, sizing, z-index and others.



These positioning aids are only present on the preview, never in the published version on the Dashboard.

Device Profiles

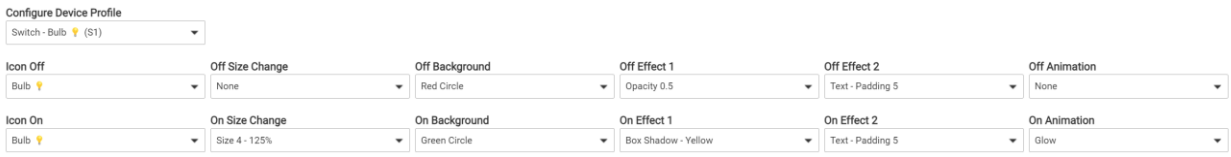
Device profiles are a collection of settings that govern how a given device will display in its various states. If we look at the device profile for “Switch – Bulb….” it looks like this.

The first row indicates all the properties of the icon when it is in an **off** state. In this case it will display a bulb, with a red circle background and an opacity of 0.5.

The second row indicates all the properties of the icon when it is in an **on** state. In this case it will display a bulb, with a green circle background. The bulb will display like this in its two different states.

Any device that is assigned this profile (in the same room) will have the same display characteristics.

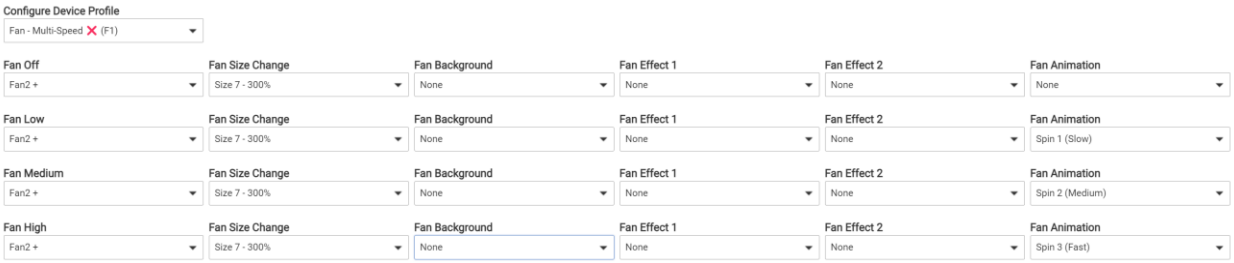
If we don’t like the defaults, we can change them very easily as shown below.



With the new settings they would now display like this. There are many customizations to choose from.

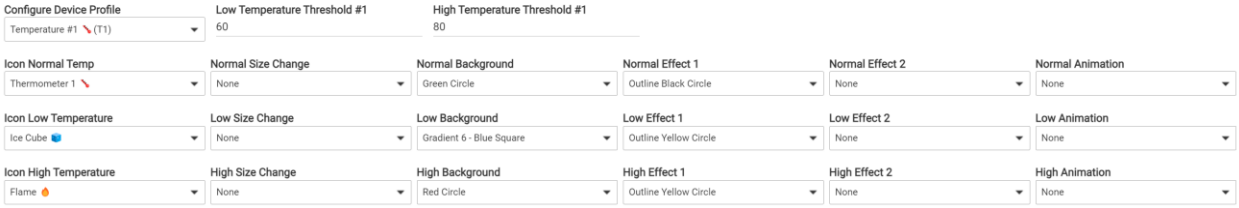
Everything about the profile is customizable so just because the name says Switch – Bulb does not mean that is it’s only use. It could be used to indicate the on/off state of any device that has the switch attribute.

Let's look at another device profile, in this case a **multi-speed fan**.

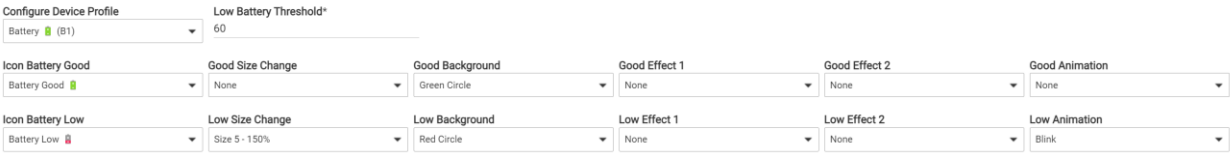


As you can see there are four definitions, one for off and another for each of the three speeds, Low, Medium and High. This time we are using an animation to spin the fan at different speeds to indicate it's state.

Some devices return numeric values vs specific states. A common example of this is **temperature** or **humidity**. In cases such as these the device profile compares the value against specified ranges and displays the appropriate icon.



A **battery** is another device where a threshold determines which icon and properties are displayed.

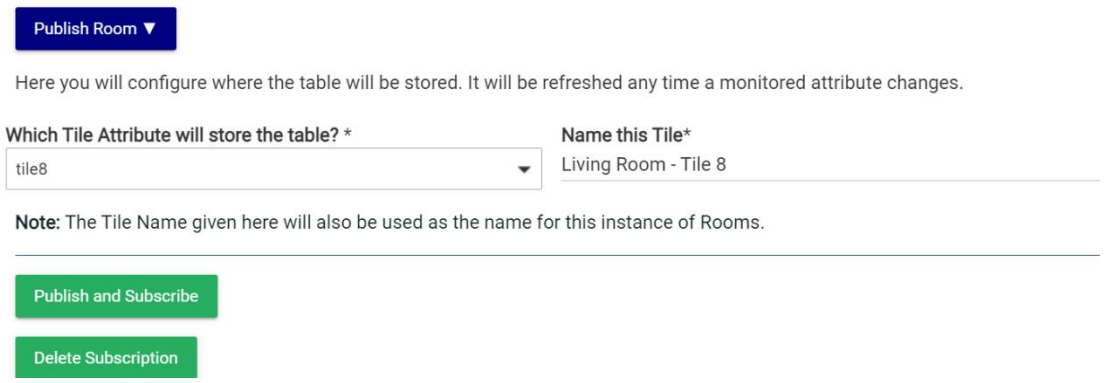


There is more to learn about device profiles, but you know enough now to be able to build effective rooms.

Publishing a Tile

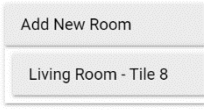
When you click on the **Publish and Subscribe** button, **Tile Builder** creates an event subscription to each of the selected devices and chosen attribute. The **Rooms** child app then remains dormant until such time as one of the monitored attributes changes. When that happens, the table is immediately 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.

The publishing options are shown below.



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 tile8 in this case.
2. **Name the Tile**. I'm going to call it **Living Room – Tile 8**. This is also the name that will be visible when looking at the **Tile Builder** parent app. I recommend you append the tile name with the tile number so you can see it on the parent screen.
3. With those values set, click on **Publish and Subscribe**.
4. Click on **Done** to close the **Attribute Monitor** app.
5. You should see your new tile listed under the **Add New Room** child app like this.



You can go back and edit this tile any time by clicking on this button. Leave your Tile Builder Rooms tab open for the moment, we will come back to it shortly.

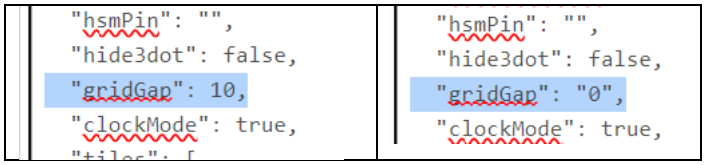
Dashboard Setup

We will now step through the process to configure a new Dashboard for our **TB Rooms**.

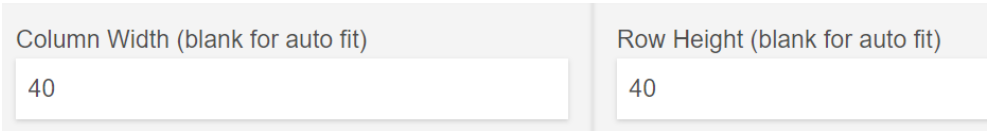
1. **Create a new Dashboard** and authorize the Tile Builder Storage Device you used earlier.
2. **Open the new Dashboard** on a separate browser tab.
3. Go back to your Tile Builder window and **click on the Classes tab.**
4. **Select and copy all the text** in the first box that starts with:

```
/* Tile Builder Section 1 - This section controls how ALL tiles display…
And ends with…
/* Tile Builder Classes End Here */
```

5. Go to your newly created Dashboard. **Click on: Gear, Advanced, CSS and paste the classes** information in here and save it.
6. Still in your Dashboard window, click on **Layout**. **Change the GridGap from 10 to "0"** as shown.

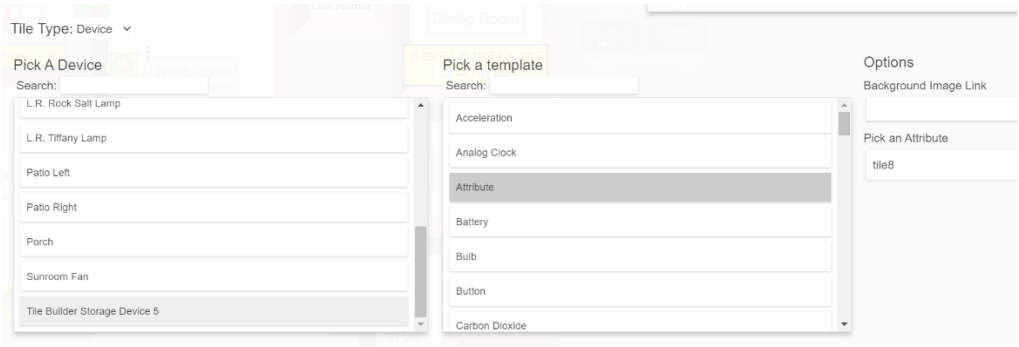


7. Still in your Dashboard window, click on the **Options** menu. **Change the Column Width and Column Height to 40** as a starting point. This smaller grid allows greater control of the proportions of the rooms you create.

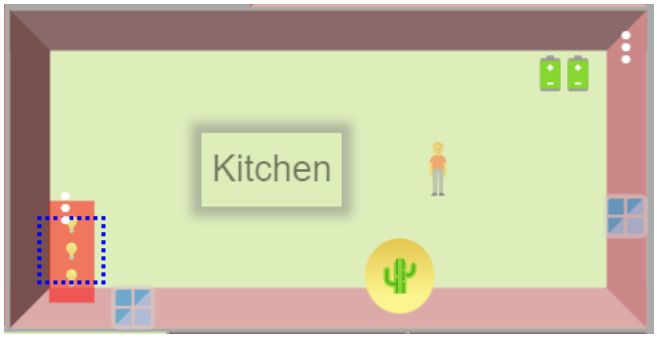


The values in steps 6 and 7 are just reasonable defaults that you can experiment with as you become more familiar with **TB Rooms**.

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



Initially when it is added to the dashboard it will be jammed into a single tile slot in the upper left corner which is only 40px square. Click on the 3 dots menu and change the size of the tile until you get the size and proportions that you want. The size you specified in the TB Rooms app only affects the preview. The final room size expands to fit the tile size that you choose.



You have now been through the entire process for creating and publishing rooms. But there is one important feature that has not been discussed.

Icon Bars

What are Icon Bars? These are groupings of icons within a single container. Icons within the icon bar do NOT change with state and are primarily designed for displaying data. For example, an HVAC system might look like this 🔥 64°F / ❄️ 74°F / 🌡️ 74°F / 💦 54% / ⚙️ Idle Displayed here are the heating setPoint, cooling setPoint, currrent temperature, current humidity and operating state. These icons will never change, but the numeric values will.

Why use Icon Bars? Icon Bars are neat and space efficient when displaying 3 or more items and they also provide logical grouping and formatting options. Displaying the 5 values shown above using individual objects would consume over 300 bytes vs the 175 bytes used.

To enable an Icon Bar click on an Icon Bar tab and set the number of devices to something other than zero.

Icon Bar Properties

Position X%, Y%: You can position the Icon Bar using X,Y coordinates just like any other object.

Text Color: The color of any text (excluding eMoji's) within the Icon Bar.

Alignment: Alignment does not work as you might initially expect. If you set an X position of 0% and the alignment to Center, it will be aligned as you expect. However, if you set the X position to 20% and the alignment to Center then the Icon bar will be perfectly in the middle of the space between 20% and

100%, in other words 10% to the right of center. If the X position were set to 10% and the Icon Bar were configured to align left, it would be aligned at 10%, not 0%.

Z-Index: This is the name given to the stacking order of objects in HTML. The default Z-Index is 0. If you want the Icon Bar to always be on top change this to a positive number. To push it to the bottom, use a negative number.

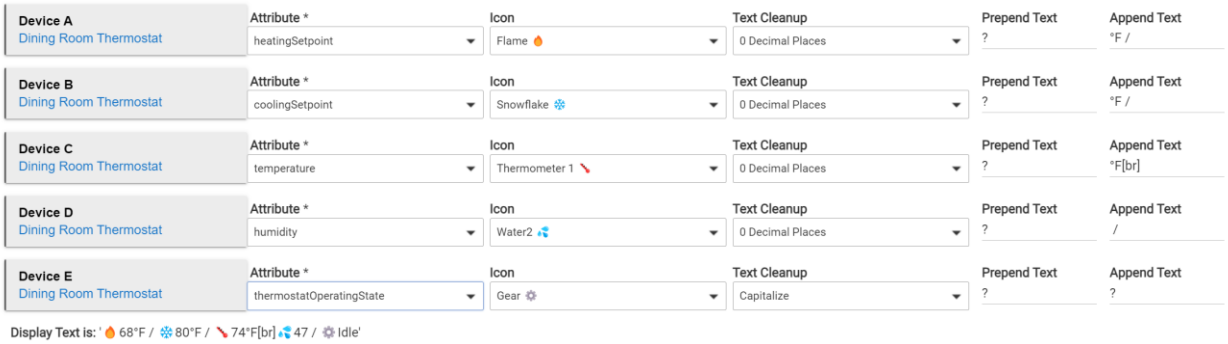
Size Change: You can make the Icon Bar smaller or larger to fit the space.

Background: You can add a background to make the Icon Bar contents stand out.

Effect1\Effect2\Effect3: These allow you to change the appearance of the Icon Bar and make them more visually interesting. Similar formatting could be used to group like information across multiple rooms.

IconBar Devices

You select the Icon Bar Devices and Attributes in the normal way, but you have a few extra options.



Icon: Icon Bar icons do not change state so this Icon represents the type of information being displayed, not the state of a device.

Text Cleanup: This provides options for making the data more presentable such as setting the number of decimal places or capitalizing text. Just choose the most relevant option.

Prepend\Append Text: You can add text and HTML tags between the various values to add units and provide formatting. In the example below the [br] tag is used to force the Icon bar onto two lines.

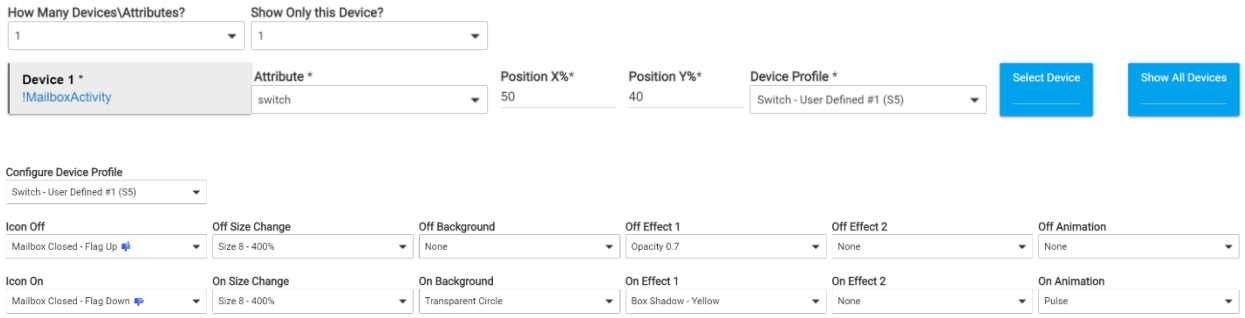
In the above example the resulting Icon Bar looked like this.

Tile Builder Rooms – Standard

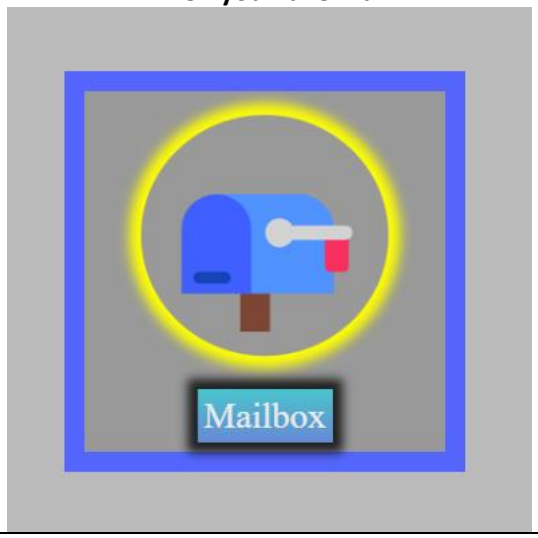
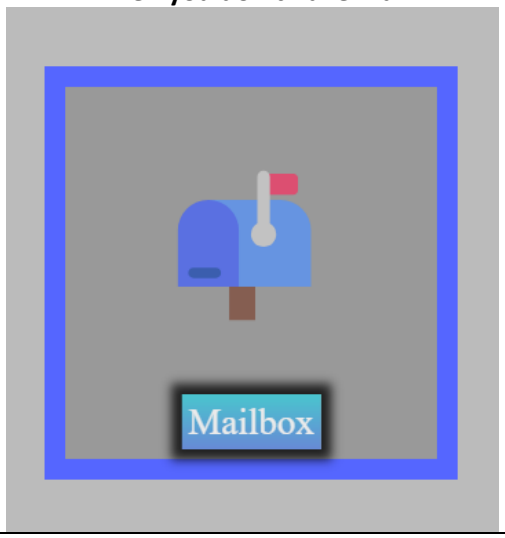
The focus of this document has been on building Rooms, but if you have the Standard (free) version, you can still get some interesting tiles despite the 3-device limit. Here we will build some sample tiles.

Mailbox Example

I have a contact sensor in my mailbox and a little RM logic for a virtual switch to indicate whether the mail has been delivered or not. In this case we only need one device.

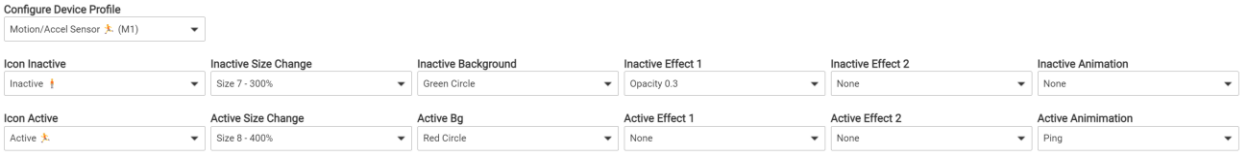


This is how the tile appears in each condition.

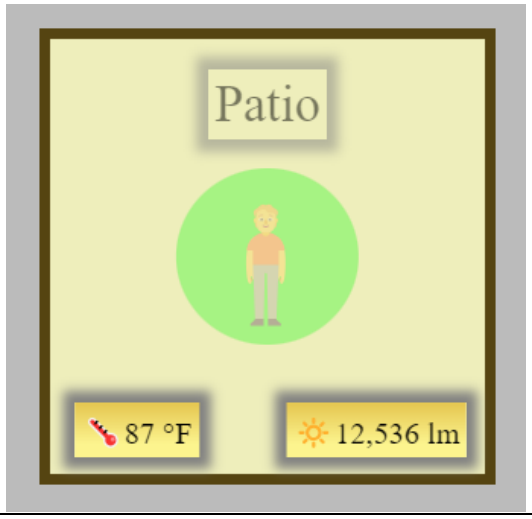
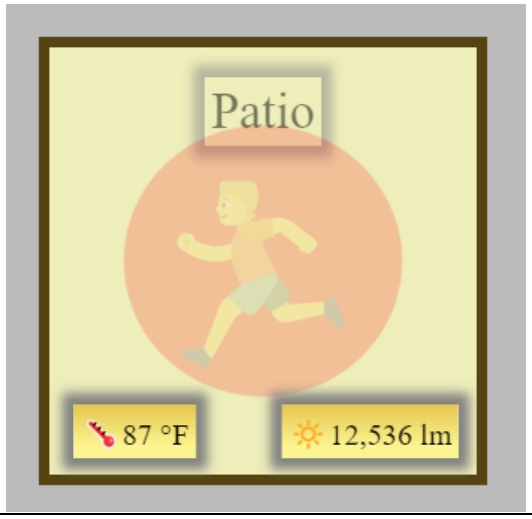
When you have mail.	When you don't have mail.
	

Motion Sensor Example

A motion sensor is a good example of a device that often has multiple attributes. In this case the active\inactive state drives the large icon in the center using the settings below.



The other two positions are filled with other attributes from the motion sensor, those being illuminance and temperature. In both these cases I chose to use a numeric device profile so that the value would be displayed.

Hue Motion Sensor - Inactive	Hue Motion Sensor - Active
	

This is what a Numeric Device profile looks like. There are 3 separate Numeric Device profiles and another 3 Text Device profiles. Because we want them to look different, I used two different numeric profiles.

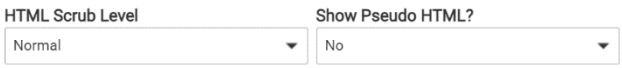
Remote Control Example

There are certain types of devices that don't have a good representation via the Hubitat dashboard. A scene controller is a good example of this. With Tile Builder this is not a problem as you can access any device\attribute combination and display those values.

Here is a simple mockup of a 3-button remote control showing the last button pressed, the health status and the battery condition.

Advanced Tab

The contents of the Advanced tab look like this.

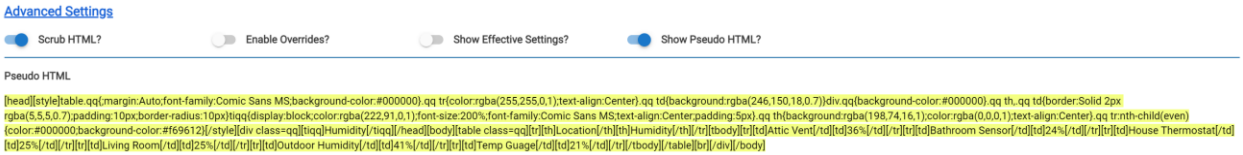


Scrub HTML

When this is enabled a scrubbing routine is employed to remove excess characters from the HTML. For example, the default text alignment is left so any references explicitly setting this value can be removed. The higher levels of scrubbing remove additional tags from the HTML. Most browsers have no issue with this but if you find your tile does not render properly then change this setting back to Basic or Normal.

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 Topics

This section covers some tips and techniques I have discovered\created as I work with Rooms. Even though I wrote **Tile Builder** I'm still finding new ways of doing things which may be of interest to others.

Creating Stacked Icons

Use the text profile and select the "Blank" icon. Enter the Icons to use with a line break [br] between them.

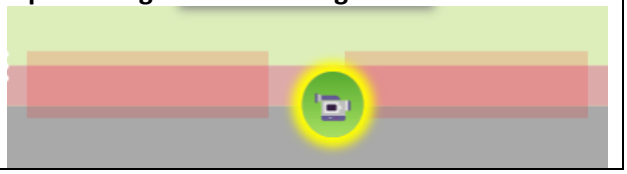
Configure Device Profile	Prepend Text	Append Text	**Outcome**
Value Text #3 (V6) Icon Text 3 Blank	[br]	[br]	Off

Creating Wall Openings

Besides icons for windows and doors there are some Unicode character strings that can be used to represent a wall opening.

Opening Extra Large
Opening Extra Large
Opening Large
Opening Large
Opening Medium
Opening Medium
Opening Small
Opening Small

By carefully sizing and positioning these strings in the right place you can create an impressive effect.
Tip: Make the background color the same as the item color to create a solid object without any gaps.

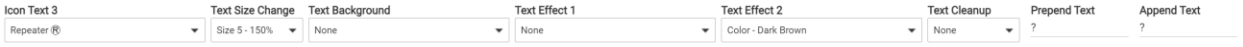
Closed garage doors.	**Open Garage Doors flashing red.**
	

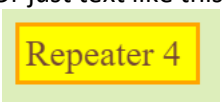
Placing No Attribute Objects

There are some devices such as dedicated repeaters which have no accessible attributes to select. Similarly, you may wish to place other objects in your Room to provide visual reference. To enable this the Tile Builder Storage Driver has a built-in empty attribute fittingly called **emptyAttribute**.

Select this Device and Attribute along with the **Value Text** device profile.

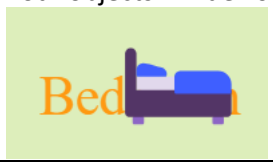
Complete the properties to get the desired effect.



The result looks like this.	But it could be this:	Or just text like this.
		

Z-Index

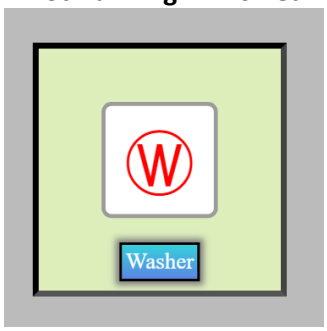
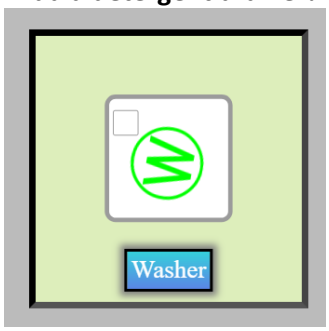
This is the HTML name for the layer an object is on. By default, an object is placed on layer 0, but sometimes two objects overlap, and we don't get the desired outcome.

Both objects Z-index 0	Title Z-index 1, bed Z-index 0
	

You can change the Z-Index of the title, Icon Bars as well as individual icons. Only use z-Index when it is required otherwise you are wasting valuable space in the 1,024-byte budget.

Compound Icons

Because Icons can overlap and each one has its own unique properties, we can use this to make compound Icons if we have the space available. There are no Icons in the emoji set that are a great substitute for a Washer and Dryer. But if we want to we can make our own.

Not Running - W is Red	Running – W is Green and Spins	Add a detergent drawer!
		

This finished tile only uses 3 device\attribute slots so it could be made with Tile Builder Standard. The final size of this Tile is 551 bytes.

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]Living Room [/u]** in the title field and the title would be displayed in underline, **[b]Living Room [/b]** and it would display in bold. Multiple HTML tags can be used.

Note: I have observed that it is not necessary to close HTML tags such as [/b] or [/u] located in text fields such as Title\Header\Keyword substitution. This can save a little space, especially in Keyword substitution where the values are repeated on each row.

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.

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

Customizing Tile Builder Rooms

In this section we will look at the different ways in which you can customize your TB Rooms to get the results you want.

Customize Base Colors

The use of green, yellow and red to indicate good, OK and bad is pretty universal. Not everyone likes the same shade of red or green. To make this easy to modify these defaults go to the CSS in your dashboard and look for a line that looks like this:

:root {--myUnderline-color:purple; --myGreen-color:lime; …. }

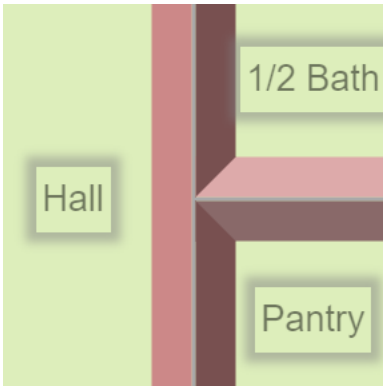
Here you can make a universal change to the color definition that TB Rooms will use throughout. You can use acceptable named colors as well as 6 digit or 8-digit hex colors with a leading # symbol. Using this value: *--myGreen-color:#228B2280* would change all the solid green colors to this semi-transparent forest green. **Note:** This setting has no effect on gradient greens.

Once you are happy with your new colors paste the whole **:root{……..}** line into the top of the **User Defined Classes** section in the Classes tab. Now it might look like this:

These new color definitions will now apply to the preview window as well as the dashboard.

Changing the Room Gap

When you have multiple rooms adjacent within a dashboard it will look something like this by default.



Notice the thin grey lines between the rooms? What you are seeing is a gap between the rooms that allows you to see through to the background. By default, this value is 2px. This value is determined by a CSS variable called **--myGridGap** which is easily adjusted to make the gap larger or make it disappear altogether.

To adjust the RoomGap, locate this section in your CSS and change the value of **--myRoomGap** to the desired amount.

```
/* Tile Builder Section 2 - Tile Classes Start Here */
:root {--myUnderline-color:purple; --myGreen-color:lime; --myRed-color:red; --myOrange-color:orange; --myRoomGap:2px}
```

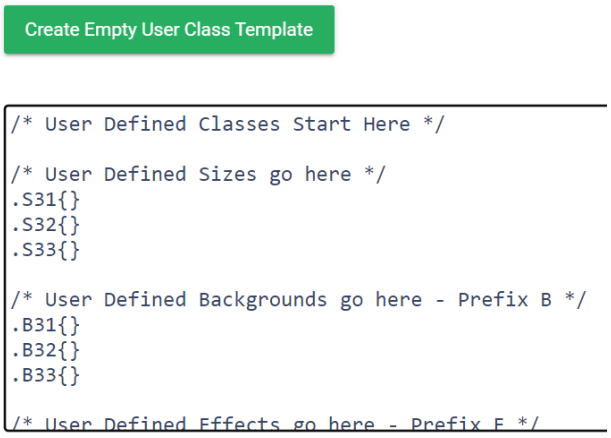
Extending the Menu Options

At the bottom of each of the Icon modifiers, Size, Background, Effects and Animations you will see **User Defined – XXX** as selectable options.

Size	Background	Effects	Animation
Size 8 - 400%	Warning Square	Z-Index 1	Spin 3 (Fast)
Size 9 - 500%	Warning Circle	Z-Index 2	Wiggle
User Defined - S31	User Defined - B31	User Defined - E31	User Defined - A31
User Defined - S32	User Defined - B32	User Defined - E32	User Defined - A32
User Defined - S33	User Defined - B33	User Defined - E33	User Defined - A33
		User Defined - E34	
		User Defined - E35	

To make these user-defined options work you must define the classes. That might sound tricky, but really, it's very easy in TB Rooms.

Go to the **Classes** tab. If the window shown below is empty, click on the **Create Empty User Class Template** and it will then look like the image below.



In this example we are going to define a different background (B) using the empty .B31{} class. How do you know what to put in here? Well, you go to the TB Rooms classes and look for the category that you are adding. This is what the Backgrounds classes section look like.

```
/* Background colors go here - Prefix B. */
.B0{display:inline-block;background:black}
.B1{display:inline-block;background:black;border-radius:50%}
.B2{display:inline-block;background:white}
.B3{display:inline-block;background:white;border-radius:50%}
.B4{display:inline-block;background:var(--myGreen-color)}
.B5{display:inline-block;background:var(--myGreen-color);border-radius:50%}
.B6{display:inline-block;background:var(--myRed-color)}
.B7{display:inline-block;background:var(--myRed-color);border-radius:50%}
.B8{display:inline-block;background:var(--myOrange-color)}
.B9{display:inline-block;background:var(--myOrange-color);border-radius:50%}
.B10{display:inline-block;background:yellow}
.B11{display:inline-block;background:yellow;border-radius:50%}
.B12{display:inline-block;background:transparent;border-radius:50%}
```

Let's say we want to add blue as a background color option. Just copy over what is used for Black and change the color to blue. It should now look like this:

```
/* User Defined Backgrounds go here - Prefix B */
.B31{display:inline-block;background:blue}
.B32{display:inline-block;background:blue;border-radius:50%}
.B33{}
```

If I try them out it looks like this:

User Defined - B31	User Defined – B32	User Defined – B32 & 15px padding
Bedroom	Bedroom	Bedroom

All you must do now is copy the **User Defined Class** block into the dashboard CSS and then it will work their also. For the sake of consistency just copy the whole block, even if some of the classes are empty.

In TB Rooms most of the defined classes focus on individual properties but you have no such constraint in **User Defined Classes,** you can any group of properties if it is legal CSS.

Example 1:

Defining a background to use a texture image.

```
.B33{background:#695100;background-image:url('https://www.transparenttextures.com/patterns/wood-pattern.png')}
```

Bedroom

Example 2:

Even though we are defining the “background” attribute B33 we are free to define any valid property here, background or otherwise.

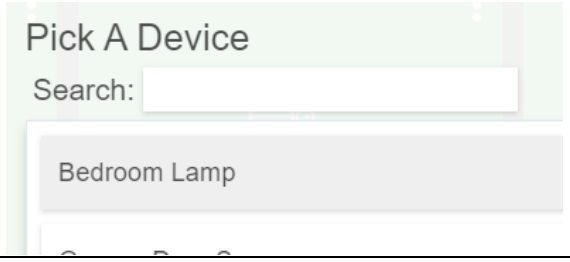
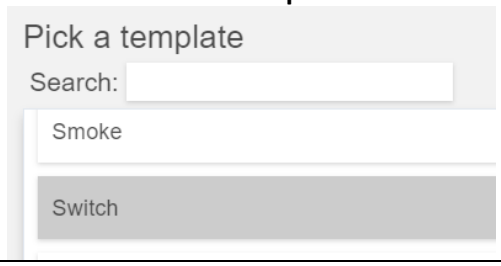
```
.B33{letter-spacing: 3px; text-shadow: 8px 11px 5px rgba(168,158,32,0.8); background-color: #282828; color:#cfc547
!important; padding: 30px}
```

Bedroom

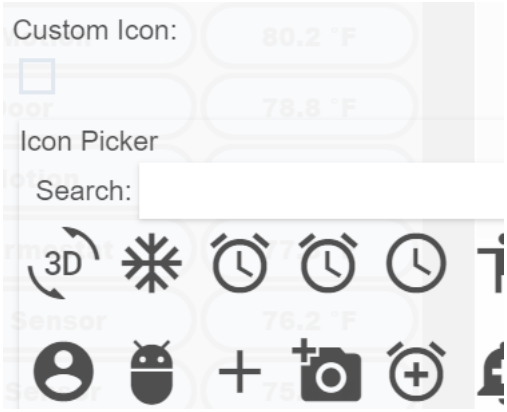
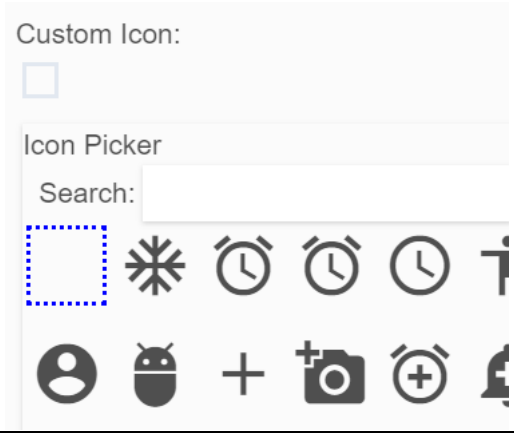
The same lessons hold true for all other **User Defined** menu options.

Adding Controls

Everything we have done so far monitors the state and values of devices and their attributes. What about control? Because we are using the native Hubitat Dashboard we can add controls in the familiar way to the dashboard.

Pick a Device:	Pick a Template
	

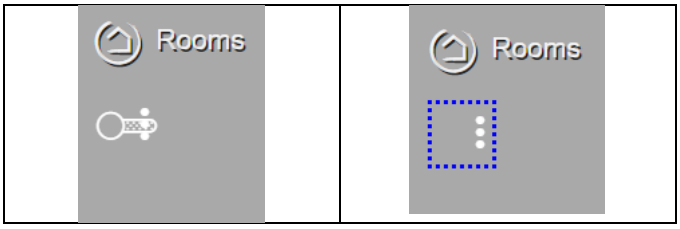
If you have already added the CSS from **TB Rooms** and you click on **Custom Icon:** your screen will look like the screen on the right with the 3D Rotation symbol replaced with a blue dotted box.

Normal Icon Picker	With TB Rooms CSS Added
	

In fact, the 3d_rotation symbol has been made transparent and a dotted border has been added to it, but the effect is the same.

Switches

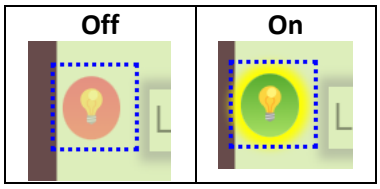
Let's add a bulb with a switch template using the above method. A small switch will appear in the upper left corner of your dashboard as shown in the first picture. If you refresh your browser, the icon will be refreshed, and the blue dotted box appears.



Remember, the grid size is only 40 x 40 so these appear quite small. Increasing the size of the tiles does not increase the size of the dotted box, that is controlled by the **Icon Size** on the **Dashboard Grid Menu.** I found 40 to be a comfortable size.

Note: Changing the icon size will also affect any "normal" Hubitat devices you place on this page.

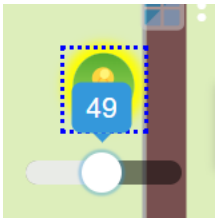
You can now reposition the blue dotted box over the top of the device to be controlled so it looks like this. Simply click on the blue box to toggle the switch.



You have multiple options to move the blue dotted box. Move it left or right on the grid, increase the height of the tile or width of the tile. Finally, go back to **TB Rooms** and fine tune the X,Y position so it is centered within the box. A little tedious but the final result is worth it.

Dimmers

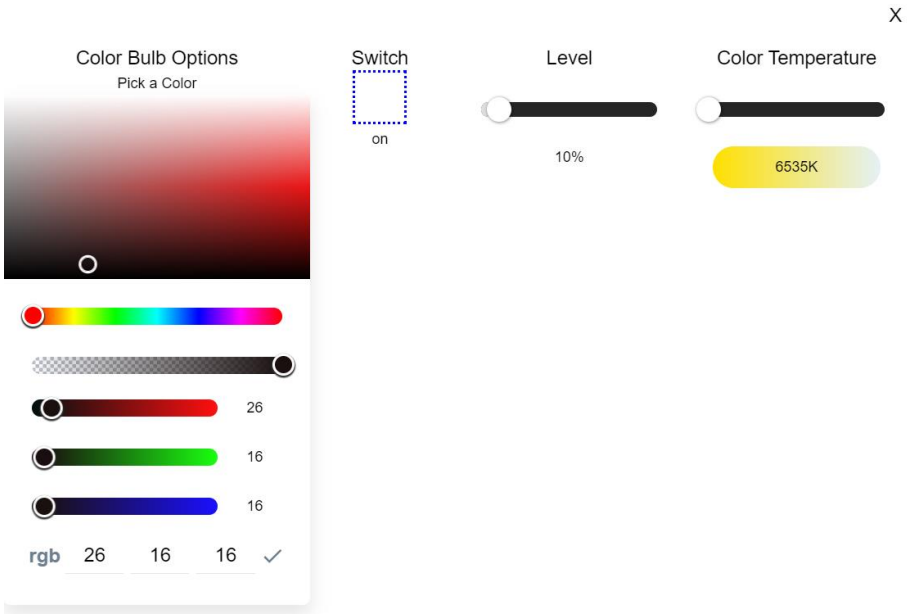
Dimmers have a switch component and a slider, but they work in the same general way as placing a switch does. The only difference is that you have a separate dimmer control that shows under the icon and operates in the same way it did before. You can use the blue dotted box to toggle the switch or use the slider to change the dimmer level.



Note: The Icon only shows off\on state, it does not change based on the dimming level, however that is indicated by the position of the slider.

Color Bulbs

Color Bulbs work somewhat differently as they have their own popup dialog. You can use the same techniques to position the blue dotted square, but when you click on the square you get the Color Bulb dialog.



The only quirk is that the icon to toggle the switch is now the blue dotted box. From here you can use the controls to make the necessary changes to the device. Note: Currently the TB Rooms icons do not show the current color\level or color temperature of the bulb, just the off\on status.

Other Controls

These same techniques will likely work for other device templates but I can't say I've spent any serious time checking them out.

It's a Wrap

Well, if you made it this far you are ready to exploit most of the power of **TB Rooms** 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.

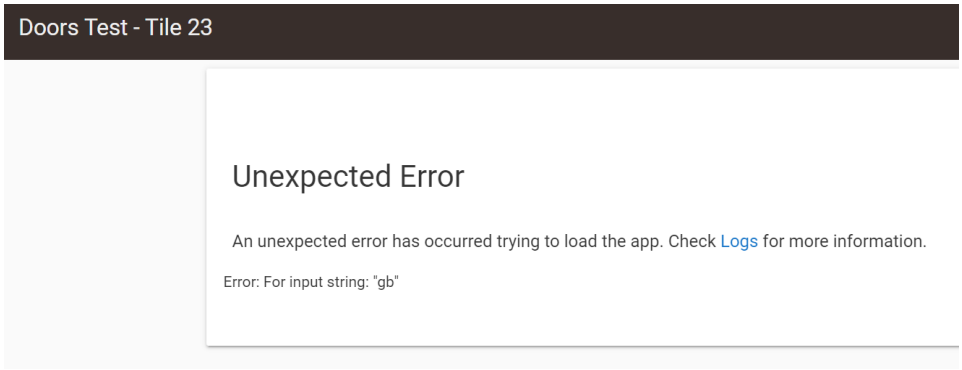
This is the fourth and probably the best **Tile Builder** module that I have written. I have ideas for several more, but whether they come to fruition largely depends on the willingness of the Hubitat community to acknowledge value in quality software and donate towards the ongoing development of the project.

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

Appendix A

Recovering Failed Tiles

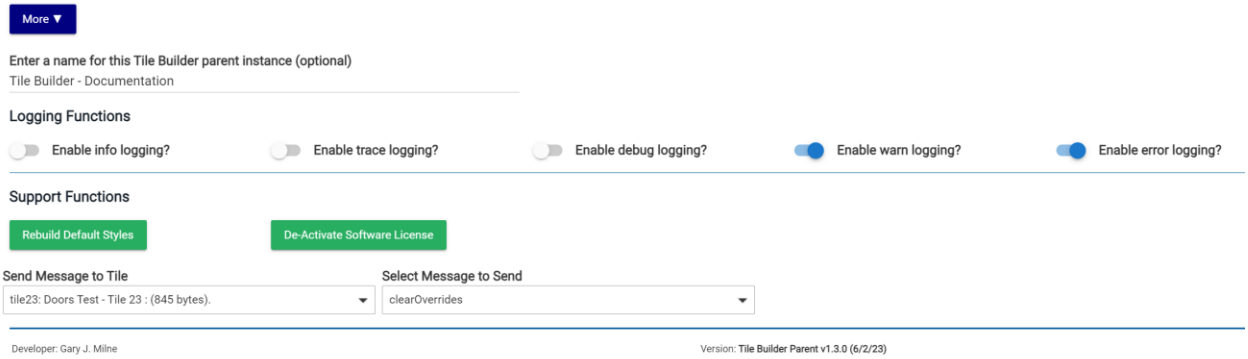
Eventually you will run into a condition where you make a change to a tile, and you get an error condition like this.



This is probably the result of a recent change creating an error condition that the program does not catch. This will most likely come about by adding a new device\attribute that has some data in an unexpected form, such as a null value, that causes the error. Another likely condition is when a device is deleted from Hubitat but still has a placeholder in TB Rooms.

It takes a while to create a tile the way you want it so if it is possible to Prior to version 1.3.0 your only option was to delete the tile and start over. With 1.3.0 you now have a good chance of recovering the tile. Here is how you do it.

Go to the Parent App and open the **More** section which will look like this.



Under **Send Message to Tile** select the problematic tile in the dropdown list.
Under **Select Message to Send** select the option that relates to the last change you made. In this case I chose **clearOverrides**.
Now go to the problematic child app and do a refresh of the browser. In this case the Overrides were cleared and now the app loaded successfully.
If the app failed to load you can try the other recovery options and repeat the process.

Once the child app has been recovered you must go to the parent app and reset the recovery options to prevent them running each time the app is refreshed.