

Tile Builder Advanced Documentation Version 1.0

Table of Contents

Table of Contents	2
Introduction	4
Highlighting	5
Keywords.....	5
Thresholds.....	6
HTML Tags.....	6
Styles	7
Apply Style	7
Save Style	8
Delete Style	8
Import\Export	9
Export a Style	9
Import Style.....	10
Advanced\Overrides	11
Scrub HTML.....	11
Show Effective Settings.....	11
Show Pseudo HTML	11
Overrides.....	11
Simple Use Case	12
Class Tags	12
Class Tag Examples.....	13
Creating Effects.....	13
Classes.....	14
Class Sample Part 1	15
Class Sample Part 2	15
Class Sample Part 3	15
Static Example.....	17


Animated Icon Examples.....	18
Spinner Example	18
A Word About Classes.....	19
Scope.....	19
Naming.....	19
Saving Space.....	19
It's a Wrap	19

Introduction

This document covers the capabilities of **Tile Builder** that are enabled in the Advanced version. Specifically, these are:

- **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 \geq , $=$, or \leq 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 several Styles built into the app. These are there primarily to spark your imagination and demonstrate the power of Styles, not polished color guides.

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 very powerful tool akin to a programming model. With **overrides** you can achieve all kinds of visual effects on a table by overriding the existing contents of a field. This allows for the creation of new classes and have those classes act on different 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 about 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 coming pages.

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 very 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.

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

Door Status	
Side	open
Front	closed
Sunroom	closed
Back	closed
Patio	closed
Garage	closed
Bedroom	closed

Size: 847 bytes

With Highlighting

Door Status	
Side	X
Front	OK
Sunroom	OK
Back	OK
Patio	OK
Garage	OK
Bedroom	OK

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 \leq , $=$, and \geq . 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
?

[▶ Highlight Notes](#)

[▶ Display Tips](#)

5 Lowest Batteries

Device	Battery
Office Thermo	66%
Bedroom Sns	69%
Temp Gauge	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

[▶ Highlight Notes](#)

[▶ Display Tips](#)

5 Lowest Batteries

Device	Battery
Office Thermo	Low
Bedroom Sns	Low
Temp Gauge	71%
Bathroom Sns	75%
Basement Sns	Good

HTML Tags

You can use embedded HTML within the text replacement fields by using `[]` braces. For example replace **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?

Apply Style

Save Current Style

Delete Selected Style

► [Styles Notes](#)

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

Humidity

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

Style: Wood

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																												
<table> <tr> <th>Location</th><th>Humidity</th></tr> <tr> <td>Attic Vent</td><td>62%</td></tr> <tr> <td>Bathroom Sensor</td><td>26%</td></tr> <tr> <td>House Thermostat</td><td>24%</td></tr> <tr> <td>Living Room</td><td>25%</td></tr> <tr> <td>Outdoor Humidity</td><td>76%</td></tr> <tr> <td>Temp Guage</td><td>24%</td></tr> </table> <p>08:10 AM</p>	Location	Humidity	Attic Vent	62%	Bathroom Sensor	26%	House Thermostat	24%	Living Room	25%	Outdoor Humidity	76%	Temp Guage	24%	<table> <tr> <th colspan="2">Humidity</th></tr> <tr> <td>Attic Vent</td><td>63%</td></tr> <tr> <td>Bathroom Sensor</td><td>26%</td></tr> <tr> <td>House Thermostat</td><td>24%</td></tr> <tr> <td>Living Room</td><td>25%</td></tr> <tr> <td>Outdoor Humidity</td><td>76%</td></tr> <tr> <td>Temp Guage</td><td>24%</td></tr> </table> <p>08:11 AM</p>	Humidity		Attic Vent	63%	Bathroom Sensor	26%	House Thermostat	24%	Living Room	25%	Outdoor Humidity	76%	Temp Guage	24%
Location	Humidity																												
Attic Vent	62%																												
Bathroom Sensor	26%																												
House Thermostat	24%																												
Living Room	25%																												
Outdoor Humidity	76%																												
Temp Guage	24%																												
Humidity																													
Attic Vent	63%																												
Bathroom Sensor	26%																												
House Thermostat	24%																												
Living Room	25%																												
Outdoor Humidity	76%																												
Temp Guage	24%																												

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 around 900 bytes 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.
<p>Save as Style: (Tab or Enter)</p> <p>My New Style</p> <p>Save Current Style</p>	<p>*Style-AM Wood</p> <p>*Style-AM-Pastel Swirl</p> <p>*Style-AM-Small For Lots of Data</p> <p>Style-AM-My New Style</p> <p>Apply Style</p>

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 possible.

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

Delete Selected Style

Import\Export

Hubitat has a vibrant community forum and it's a great asset, so why not use it. When **Tile Builder Advanced** users come up with 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 style. How great is that!

Paste Basic Settings Here!
?

Paste Overrides Here!
?

Import Style

Clear Import

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.

Create a new Topic

My Great New Style

Get H Apps optional tags

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%)]

Import Style

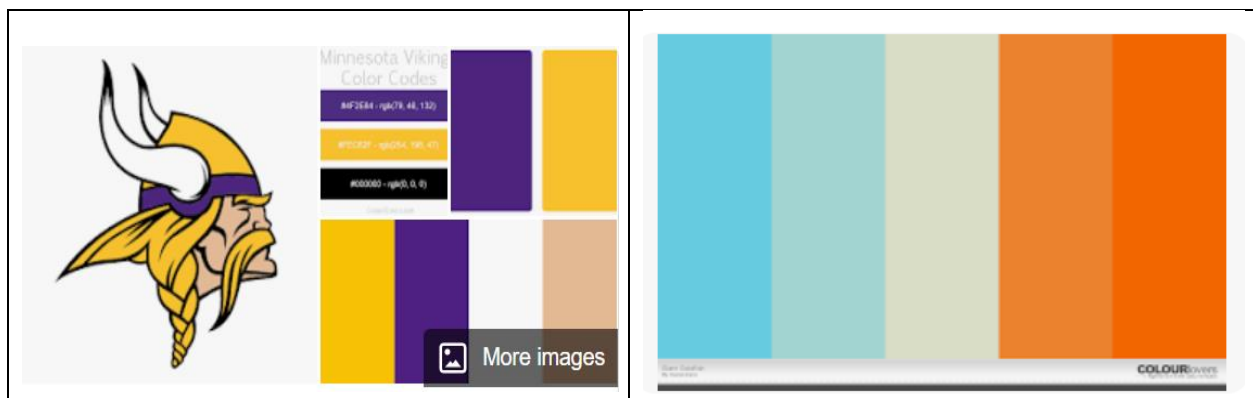
Clear Import

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 an hour 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.



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#Separate, #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, #hpf#0, #hta#Center, #htc#rgba(0,0,0,1), #hto#1, #hts#100, #hts1#125, #hts2#125, #ifFrameColor# #B8686, #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, #rabc#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# #f23516, #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#, #tof#1, #top1#1, #top2#3, #tp#5, #ts#200, #tt#Humidity, #tr1#?, #tr2#?, #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][table class=qq[tr][th]Location/th][th]Humidity/th][tr][tbody][tr][td]Attic Vent[td][td]36%/td][tr][td]Bathroom Sensor[td][td]24%/td][tr][td]House Thermostat[td][td]25%/td][tr][td]Living Room[td][td]25%/td][tr][td]Outdoor Humidity[td][td]41%/td][tr][td]Temp Gauge[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 by some value 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 probably 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.

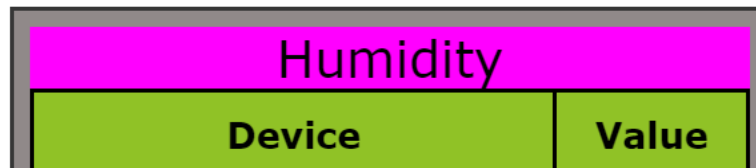
Class Tags

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%

Produces this rather ugly result:



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;
#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 two remaining tags **#Class#** and **#Box#** are 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 help get you started with **Overrides** a helper has been built in with about 40 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, there is nothing else at play.

Let's 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.

Humidity

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

13:47 PM

Edit the **Settings 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

13:54 PM

Many 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>.

Classes

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

Let's 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.

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

Preview

Code

Copy this to the element you want to animate.

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

Copy

Copy this after the above selector.

```
@keyframes myAnim {  
  0%,  
  50%,  
  100% {  
    opacity: 1;  
  }  
  25%,  
  75% {  
    opacity: 0;  
  }  
}
```

Copy

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.

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 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 called using simply `"cl"`. This name can be anything, but shorter names save precious space.


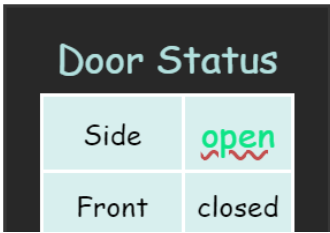


```
#Class#=.cl{background-color: #ff0000;}
```

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	Highlight 2 Color (#1b570a)	Highlight 2 Text Scale
<input type="text" value="#0ce486"/>	<input type="text" value="125"/>	<input type="text" value="#1b570a"/>	<input type="text" value="125"/>
<input checked="" type="checkbox"/> Enable Keyword #1	Enter Keyword #1	Replacement Text #1	
	open	[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:

<pre>#Class#=.cl{background-color: #ff0000;}</pre> 	<pre>#Class#=.cl{text-decoration: underline wavy #C34E4E;}</pre> 
<pre>#Class#=.cl{outline: 2px solid red;}</pre> 	<pre>#Class#=.cl{box-shadow: 0px 0px 10px 10px #E8DD95;}</pre> 

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 it's normal position.

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

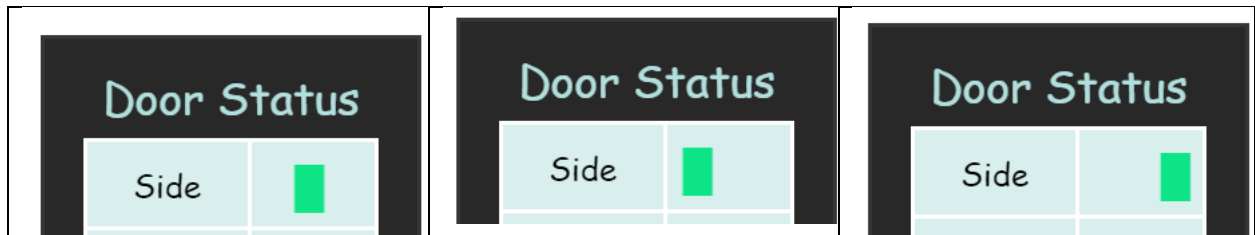
We can use it as a class by doing this in overrides. The class name is cl1.

```
#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.

<input checked="" type="checkbox"/> 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.



Again, you don't have to write most of these statements by hand. Use the **Overrides Helper** examples or an HTML\CSS generator such as <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 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 or switches.



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 these naming collisions. If you have two classes by the same name anywhere in the dashboard only one of them will work.

Saving Space

Because classes are global in scope they can also be defined in the dashboard CSS interface like this animation for motion detectors that was discussed previously.

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

Save CSS

The above class used to indicate motion is implemented via highlighting using a replacement phrase of **[div class=cl]active[/div]** .

I've tried to make **Tile Builder** accessible to all **Hubitat** users and have avoided any mention of directly editing 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 the tile to dashboard CSS code and keeping everything in one place.

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 ideas for two 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. The more money I raise doing this the more time I can devote to its future development.

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	#hpt#=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.