Tile Builder
Grid
Version 2.0

Revised 4/23/24

Table of Contents

Overview

Previous versions of Tile Builder have limited the user to two columns of data. The left-hand column for the device name and the right-hand column for the data. Tile Builder Grid allows the user to create tables up to 5 columns wide and place text and data anywhere they wish within that Grid. For the purposes of this document, I'm going to assume the reader is already familiar with using one of the other Tile Builder apps and knows how to perform the normal table editing functions.

Tile Builder Grid has two modes of operation.

Mode: Device Group

In this mode the user selects a group of devices they are interested in and fills out a one-line template for how the resulting data will look. That one-line template is then applied to each device in the device group to create the final table. In this mode the information is always sorted by the device label.
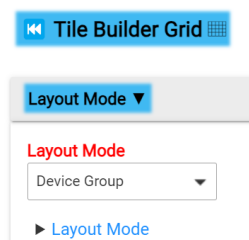
Mode: Free Form

In this mode the user must create a template for the entire table. This mode is used for non-repeating data such as a weather tile, hub info tile or any kind of blended information. That template is then filled with the data for publication. In this mode the information layout always remains the same.
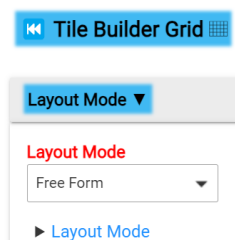
Layout: Device Group

Let's say I want a Table that shows me Temperature, Humidity and Battery level for a range of devices. If I create a new Grid it will look like this when I launch it.

We will start with a simple case and put each piece of data in its own column. So, we will need 4 columns and 3 variables (temp, humidity, and battery). The screen would look like this.

Now we must place our variables within the template as shown below.

All we must do now is select the devices we want to display within the table. If we are only dealing with a single capability, we can filter the list of devices by that specific capability using the **Select Filter (Optional)** control. If you want to select multiple capabilities do not use the Select Filter. Once I select my devices the table will look something like this.

Device	State	Other 1	Other 2
Bedroom Sensor	68.51	N/A	43
Dining Room Thermostat	68.0	35	N/A
Hub Info	88.2	N/A	N/A
Kitchen Motion Sensor	62.55	N/A	75
Living Room Sensor	65.68	41.3	100
Patio Motion Sensor	47.14	N/A	100

You can see that some fields have N/A indicating that the device does not have that attribute. The table looks O.K. but needs a little cleanup. If you have used Multi-Attribute Monitor, then the next few steps will be familiar. I'm going to shorten the device name, add units, and change the reported values to 0 decimal places like this:

Variable (%temperature%)	Cleanup	Rules	Variable (%humidity%)	Cleanup	Rules	Variable (%battery%)	Cleanup	Rules
temperature	0 Decimal Places	None	humidity	0 Decimal Places	None	battery	0 Decimal Places	None

▶ Cleanup and Rules Help

Gather Device Details	Default Date Time Format	Truncate Device Name\Label	Invalid Attribute String
deviceLabel, deviceName, lastActivity	To: HH:mm	Second Space	N/A

Template Column 1	Template Column 2	Template Column 3	Template Column 4
%deviceLabel%	%Temperature% °F	%humidity%%	%Battery%%

Here is the result of those changes.

Device	State	Other 1	Other 2
Bedroom Sensor	69 °F	N/A%	43%
Dining Room	68 °F	35%	N/A%
Hub Info	88 °F	N/A%	N/A%
Kitchen Motion	63 °F	N/A%	75%
Living Room	66 °F	41%	100%
Patio Motion	47 °F	N/A%	100%

06:36 AM

Better, but still room for improvement. We need some better headers; shrink the table to recover space and I don't like the N/A%. I'm going to assume you know how to fix the first two items yourself. The third can be changed using the **Invalid Attribute String** field.

To finish it off I changed the Style to "Marooned", disabled overrides, turned off the Title and reduced the table width and the result looks like this:

Device	Temp	Humidity	Battery
Bedroom Sensor	69 °F	-- %	43 %
Dining Room	68 °F	35 %	-- %
Hub Info	88 °F	-- %	-- %
Kitchen Motion	63 °F	-- %	75 %
Living Room	66 °F	41 %	100 %
Patio Motion	47 °F	-- %	100 %

Not bad for about 2 minutes of work. The size of this table is 850 bytes but reduces to 671 bytes with Aggressive scrubbing.

Device Details

Device details are information about a device that are not device attributes. Examples of these are the device name or label, the room it is in, the last activity date or any past event information that is retrieved. In version 1.X of Grid a limited set of these were retrieved automatically and could be placed into the template using the %variable% syntax.

In Grid 2.0 the number of accessible device details has expanded and to maximize efficiency these are now individually selectable. The screenshot below shows a partial list of available details.
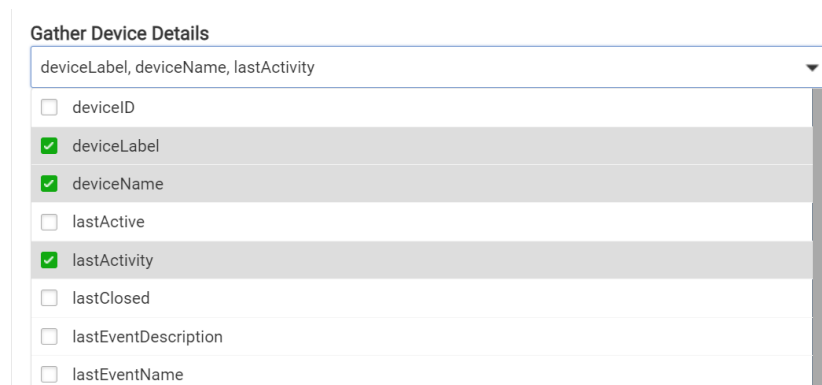
For a complete listing of available device details expand the "**Text Fields and Variables Help**" as shown:

▼ Text Fields and Variables Help

Italicized Fields: Any field/control with an italicized title does not automatically refresh the table when the con

Variables: You can place a variable in any text field using the syntax %variableName%. Variable names are as

Units: Common units can be cut and paste from here: °F °C

Restricted Characters: Tile Builder is sensitive to the presence of special ASCII characters like () [] {} \ , " * + ?

▶ Built-In variables - Any Layout Mode

▼ Built-In Variables - Device Group Mode
- %deviceName% - Name of the device.
- %deviceLabel% - Label of the device.
- %deviceID% - The numeric ID of the device.

▼ Selectable Device Properties - Any Layout Mode
- %deviceName% - Name of the device.
- %deviceLabel% - Label of the device.
- %deviceID% - The numeric ID of the device.
- %lastOn% - Last time 'switch' was turned on. N/A if not applicable or not available.
- %lastOff% - Last time 'switch' was turned off. N/A if not applicable or not available.
- %lastOpen% - Last time 'contact' sensor was opened. N/A if not applicable or not available.
- %lastClosed% - Last time 'contact' sensor was closed. N/A if not applicable or not available.
- %lastActive% - Last time 'motion' sensor was active. N/A if not applicable or not available.
- %lastInactive% - Last time 'motion' sensor was inactive. N/A if not applicable or not available.
- %lastLocked% - Last time 'lock' was locked. N/A if not applicable or not available.
- %lastUnlocked% - Last time 'lock' was unlocked. N/A if not applicable or not available.
- %lastPresent% - Last time 'presense sensor' was marked present. N/A if not applicable or not available.
- %lastNotPresent% - Last time 'presense sensor' was marked not present. N/A if not applicable or not available.
- %roomName% - The room a device is associated with. N/A if not applicable or not available.
- %lastActivity% - Date and time of Last Activity on the device as indicated by the lastActivityAt device property. N/A if no
- %lastEventName% - Name of the last attribute that changed. N/A if not applicable or not available.
- %lastEventValue% - Value of the last attribute that changed. N/A if not applicable or not available.
- %lastEventDescription% - The description text associated with the last change. N/A if not applicable or not available.
- %lastEventType% - The type of the last command that was received. N/A if not applicable or not available.
- **ALL Device Properties are instantaneous and are ONLY calculated when the table is refreshed.**

Device Details Example 1:

Let's take the example of a sprinkler system where we want to be able to check on **is it running**, **when did it last run** and **how long did it run for**. For this we will need four variables, A) device name, B) switch state, C) lastOn and D) lastOnDuration. Of these, only the Switch is an attribute, the others are all device details.

Note: My system uses switches. If you use valves, you will use lastOpen and lastOpenDuration instead.

This is how my configuration screen looks:

Add some headers and the resulting table looks like this.

Zone	Last On	Duration
1 - Front Lawn North	5:00 AM	0h 24m 59s
2 - Front Lawn South	5:30 AM	0h 25m 0s
3 - Back Lawn	6:00 AM	0h 9m 59s
4 - Flower Beds	6:30 AM	0h 25m 0s
5 - Patio West	7:00 AM	0h 9m 59s
6 - Patio East	6:31 PM	0h 1m 15s

That is great information but how about showing which zones are currently active? We are already gathering switch information so we could just add %switch% to the template like this.

But we could make it a little nicer if we used a Highlight rule to show when a zone is currently active. Here is how to do that.

Change your template line to look like this:

Template Column 1	Template Column 2	Template Column 3
%deviceLabel%	%lastOn%	%switch%

Add a rule to the switch to process all keywords.

Variable (%switch%)	Cleanup	Rules
switch ▼	None ▼	All Keywords ▼

A switch will only have one of two values, **on** or **off** so we need two keyword rules like this.

Highlights

Show Keywords ▼	How Many Keywords?
	2 ▼

Keyword Match Type	Keyword #1	Replacement Text #1
Value Contains Keyword (Ignore Case) ▼	on	[mark]%lastOnDuration%

Keyword Match Type	Keyword #2	Replacement Text #2
Value Contains Keyword (Ignore Case) ▼	off	%lastOnDuration%

Those changes result in the table looking like this. Inactive zones (off) show the duration in red whereas an active zone has the duration marked in yellow. This table was 773 bytes.

Zone	Last On	Duration
1 - Front Lawn North	5:00 AM	0h 24m 59s
2 - Front Lawn South	5:30 AM	0h 25m 0s
3 - Back Lawn	6:00 AM	0h 9m 59s
4 - Flower Beds	7:26 AM	0h 5m 59s
5 - Patio West	7:00 AM	0h 9m 59s
6 - Patio East	7:30 AM	0h 1m 49s

Note: <mark> (entered as [mark]) is a standard HTML tag used to highlight some text. The end of a table cell creates an implied closure for most tags. It is often not necessary to provide a closing tag.

With some additional changes my final Sprinkler Monitor tile looks like this and was 972 bytes.

Zone	Last On	Duration
💧 1 - Front Lawn North	Tue 6:32 PM	🔴 0h 0m 3s
💧 2 - Front Lawn South	Tue 6:33 PM	🔴 0h 0m 2s
💧 3 - Back Lawn	Tue 6:33 PM	🔴 0h 0m 3s
💧 4 - Flower Beds	Tue 7:56 PM	🟡 0h 3m 47s
💧 5 - Patio West	Tue 6:02 PM	🔴 0h 28m 44s
💧 6 - Patio East	Tue 6:31 PM	🔴 0h 1m 15s

Why do I keep mentioning the size of the tile in bytes? Because a tile that is less than 1,024 bytes will always display on the dashboard in the Hubitat app whether you are local or remote. Once a tile exceeds 1,024 bytes it is stored as a file on the hub and is only accessible on the LAN or with a VPN.

That is all you need to get started with a **Device Group,** but there is plenty more to learn in the following pages for using Grid in **Free Form** mode.

Device Details Example 2:

In this table the device details being gathered relate to contact sensors. The one highlighted in yellow is currently open. The remainder are closed.

A device that shows N/A indicates that there was insufficient data from the these devices. This can occur on inactive devices as Hubitat purges old events so Tile Builder can no longer identify when the open\closed events happened.

Contact Sensors	Last Opened	Last Closed
Garage Door Sensor (3h 49m 18s)	Last Opened: Thu 7:40 AM for 0h 0m 57s	Last Closed: Thu 7:41 AM for 3h 49m 18s
Office Door (1d 1h 20m 12s)	Last Opened: Wed 10:10 AM for 1d 1h 20m 12s	Last Closed: Mon 12:53 PM for 1d 21h 17m 20s
Side Door (1h 6m 21s)	Last Opened: Thu 10:24 AM for 0h 0m 12s	Last Closed: Thu 10:24 AM for 1h 6m 21s
~Back Door (3h 50m 12s)	Last Opened: Thu 7:40 AM for 0h 0m 8s	Last Closed: Thu 7:40 AM for 3h 50m 12s
~Bathroom Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Bedroom Deck Door (5d 20h 42m 31s)	Last Opened: Fri 2:42 PM for 0h 5m 30s	Last Closed: Fri 2:48 PM for 5d 20h 42m 31s
~Dining Room North Window (11d 23h 48m 1s)	Last Opened: Sat 11:14 AM for 0h 27m 57s	Last Closed: Sat 11:42 AM for 11d 23h 48m 1s
~Dining Room West Window (3d 14h 49m 9s)	Last Opened: Sun 11:25 AM for 9h 16m 3s	Last Closed: Sun 8:41 PM for 3d 14h 49m 9s
~Front Door (18h 50m 4s)	Last Opened: Wed 4:40 PM for 0h 0m 18s	Last Closed: Wed 4:40 PM for 18h 50m 4s
~Garage Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Hall Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Kitchen East Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Kitchen North Window (11d 23h 4m 37s)	Last Opened: Sat 10:06 AM for 2h 19m 55s	Last Closed: Sat 12:26 PM for 11d 23h 4m 37s
~Living Room East Window (9d 22h 12m 25s)	Last Opened: Mon 12:11 PM for 1h 7m 19s	Last Closed: Mon 1:18 PM for 9d 22h 12m 25s
~Living Room West Window (10d 22h 13m 3s)	Last Opened: Sun 12:32 PM for 0h 45m 11s	Last Closed: Sun 1:17 PM for 10d 22h 13m 3s
~Patio Door (2d 1h 36m 2s)	Last Opened: Tue 9:54 AM for 0h 0m 5s	Last Closed: Tue 9:54 AM for 2d 1h 36m 2s
~Patio Left Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Patio Right Window (N/A)	Last Opened: N/A for N/A	Last Closed: N/A for N/A
~Sunroom Door (15h 14m 6s)	Last Opened: Wed 5:35 PM for 2h 41m 27s	Last Closed: Wed 8:16 PM for 15h 14m 6s

11:30 AM

Layout: Free Form

Free Form mode is used to present a lot of unique pieces of data. Unlike a Device Group, the data does not repeat with each row.

To create a Free Form layout, we must first change the Layout mode as shown below.

First, we are asked to configure the variables. Each variable is just a reference to an attribute and using the UI you will connect the variable to the attribute of interest. You can define up to 40 variables in addition to multiple built-in variables described later. For this example, I'm going to create a very simple Weather Tile using the Open Weather driver as the source.

The data I'm interested in is as follows. Today: current feel, wind, forecast high\low temp and forecast. Tomorrow: predicted high\low temp and forecast. That is a total of 8 variables.

Because all this data is coming from my OpenWeather device I'm going to make that my **Default Device**. This simplifies the process of picking the device\attribute combination which can be tedious if you are selecting a lot of data. I have a widescreen monitor so I changed my **Variable Columns** to 2 and make it easier to view.

Here is my filled-out Variables section. Notice the variables and their values display at the bottom of the section as I proceed to fill it out. I have everything I need for this example so I can move on.

You can collapse the **Layout Mode** and **Configure Variables** sections to give yourself a little more room. Open the **Grid Layout** section and configure the **Data Rows** to be 2 and **Data Columns** to be 3. Your screen will look like this.

Layout Mode ▶	↙

Configure Variables ▶	↙

feelslike: 25, forecastHigh: 37, forecastHigh1: 41, forecastLow: 31, forecastLow1: 32, icon: https://tinyurl.com/icnqz/26.png, icon1: https://tinyurl.com/icnqz/28.png, windspeed: 12,

Grid Layout ▼	↖

How Many Data Rows?	How Many Data Columns?	Auto Refresh Table on Save?
2 ▾	3 ▾	False ▾

R1C1	R1C2	R1C3
?	?	?
R2C1	R2C2	R2C3
?	?	?

▶ Text Fields

I'm going to use column 1 for the text labels, column 2 for today's data and column 3 for tomorrow's data. I can now place my variables like this.

How Many Data Rows?	How Many Data Columns?	Auto Refresh Table on Save?
2 ▾	3 ▾	False ▾

R1C1	R1C2	R1C3
Forecast	%icon%	%icon1%
R2C1	R2C2	R2C3
Temperature High\Low	%forecastHigh% \ %forecastLow%	%forecastHigh1% \ %forecastLow1%

Device	State	Other 1
Forecast	https://tinyurl.com/icnqz/26.png	https://tinyurl.com/icnqz/28.png
Temperature High\Low	37\ 31	41 \ 32

We have most of the raw information in there except for Feel and Wind. We will add those later but let's dress this up a bit first including the headers.

Here we have added some units as well as some bold tags.

How Many Data Rows?	How Many Data Columns?	Auto Refresh Table on Save?
2 ▾	3 ▾	False ▾

R1C1	R1C2	R1C3
[b]Forecast[/b]	%icon%	%icon1%
R2C1	R2C2	R2C3
[b]Temp High\Low[/b]	%forecastHigh%°F \ %forecastLow%°F	%forecastHigh1%°F \ %forecastLow1%°F

Added some header information like this:

Heading 1		Heading 2		Heading 3	
Weather	💾	%today%	💾	%tomorrow%	💾

Now our table looks like this:

Weather	Thursday	Friday
Forecast	https://tinyurl.com/icnqz/26.png	https://tinyurl.com/icnqz/28.png
Temp High\Low	37°F \ 31°F	41°F \ 32°F

To get the actual weather icon to display as an image instead of a string we need to go back and apply a cleanup rule to force this change. In this case we tell it to treat the result as an image URL.

And the table looks like.

Weather	Thursday	Friday
Forecast		
Temp High\Low	37°F \ 31°F	41°F \ 32°F

Add a few more tags like this:

How Many Data Rows?	How Many Data Columns?	Auto Refresh Table on Save?
2	3	False

R1C1	R1C2	R1C3
[b]Forecast[/b]	%icon%[br][b]%today%	%icon1%[br][b]%tomorrow%

R2C1	R2C2	R2C3
[b]Temperature	High: [b]%forecastHigh%°F[/b][br]Low: [b]%forecastLow%°F	High: [b]%forecastHigh1%°F[/b][br]Low: [b]%forecastLow1%°F

…..and merge the column 2 and 3 headers.

Note: The images presented using Open Weather are a fixed size and do not scale.

Almost there, just need to add the Feels Like temperature and the Wind Speed. I'm going to add them to the header like this.

Heading 1	Heading 2
Feel: %feelsLike%°F	recast (%windspeed% m.p.h.)

This is the result.

Feel: 24°F	Minneapolis 2 Day Weather Forecast (14 m.p.h.)	
Forecast	Thursday	Friday
Temperature	High: **37°F** Low: **31°F**	High: **41°F** Low: **32°F**

 Of course, this is just a simple exercise to demonstrate the basic principles. My actual weather tile is more complex and looks like this:

Feel: 24°F	Minneapolis 3 Day Weather *No current weather alerts for this area*		Current: 33°F
Updated @ 12:17 PM on Thursday	Thursday Overcast clouds	Friday Broken clouds	Saturday Scattered clouds
Low \ High	31°F \ 37°F	32°F \ 41°F	29°F \ 34°F
Wind	14 gusting to 14	North-Northwest (340°)	↑
Lux: 6,400 UV: 0.89	Sunrise: 07:50 Sunset: 16:39	Humidity: 83% DewPoint: 29%	Rain Today: 0.0 Rain Tomorrow: 0.0

You now have all the basics you need to start creating sophisticated tables. However, there is still a lot to learn, and the next section covers the various sources of data that can be presented in the table.

Using Alternate Data Sources

Thus far we have only used the Default Device as all the data we were accessing was from a single OpenWeather device. In this section we will look at some of the other sources of data you can use.

There are 5 different options for pulling data from Hubitat.

Default Device: This is used as a quick method to access a device without having to go through the device picker each time. Use this if your table contains a lot of data from one device.

Device Attribute: This lets you select any attribute of any device on your Hubitat Hub and place it onto your table.

Device Detail: This lets you access non-attribute data from a device. This is the same data that was accessed using a Device Group, but in this case is only for a single device.

Hub Property: Hub properties are things like the Hub Name, HSM status, Mode, Sunrise, Time etc.

Hub Variable: Use this to access the values of any Hub Variables that you have created, even those that do not have a device connector.

Device Attributes

When you select a device attribute as the input you will be prompted to enter a variable name and then select a device and attribute combination. You can mix and match all kinds of data, but each one must be assigned a unique name using the Var Name field as shown below.

In Tile Builder Grid, the table on the dashboard will automatically refresh any time one of the device attributes changes.

Device Details

These fall into two categories. Those which are actual device properties and those which are derived from the event history. Device properties are things like deviceName, deviceID and roomName.

Derived values all begin with the word lastXXX, such as lastOn, lastOpen, lastPresent, lastClosedDuration and are collected by looking in the event log of a device for the specified event(s).

In this example we retrieve the last time that the Front Door was locked. Because this is a date\time value it can be formatted to your liking using the appropriate cleanup as shown below. Device Details will only refresh when the table is refreshed.

Anything that ends with the word "**Duration**" analyzes both binary values (on/off, locked/unlocked, active/inactive etc.) and calculates the amount of time it was in the requested state.

In Free Form mode, when placing a device detail onto the grid you reference the information by using whatever **Var Name** you assign to it in the UI, in this case it would be **%Duration%**.

Note: *Device Details are not subscription based. They only refresh when the table is refreshed.*

Hub Properties

These are the API retrievable properties of the Hub, for things like sunrise, sunset, HSM mode etc.

Note: Hub Properties are not subscription based. They only refresh when the table is refreshed.

Hub Variables

This option was available in Grid 1.0 but has been improved to handle all data types.

Source #1 | Var Name | Hub Variable
Hub Variable | myVar | varBool

▶ Cleanup and Rules Help

myVar: false,

Note: Hub Variables <u>do</u> use subscriptions. The table will refresh anytime one of the included Hub variables is changed.

Built-In Variables

Grid 2.0 supports an expanded list of built-in variables as shown below. These can also be found within the application under "**Text Fields and Variables Help"** as shown below.

▼ Text Fields and Variables Help

Variables: You can place a variable in any text field using the syntax %variableName%. Va

Units: Common units can be cut and paste from here: °F °C

Restricted Characters: Tile Builder is sensitive to the presence of special ASCII character:

▼ Built-In variables - Any Layout Mode
- %day% - Day of week in form: Fri
- %date% - Date in form: 22-12
- %date1% - Date in form: Dec-22
- %sunrise% - Time in form: 06:47 AM
- %sunrise1% - Time in form: 06:47
- %sunrise2% - Time in form: 6:47 AM
- %sunset% - Time in form: 21:47 PM
- %sunset1% - Time in form: 21:47
- %sunset2% - Time in form: 9:47 PM
- %time% - Time in form: 23:35 PM
- %time1% - Time in form: 23:35
- %time2% - Time in form: 11:35 PM
- %today% - Current day as day of week in form: Friday
- %tomorrow% - Tomorrow as day of week in form: Saturday
- %dayAfterTomorrow% - Day after tomorrow as day of week in form: Sunday
- **ALL Built-In Variables are instantaneous and are ONLY calculated when the table is refreshed.**
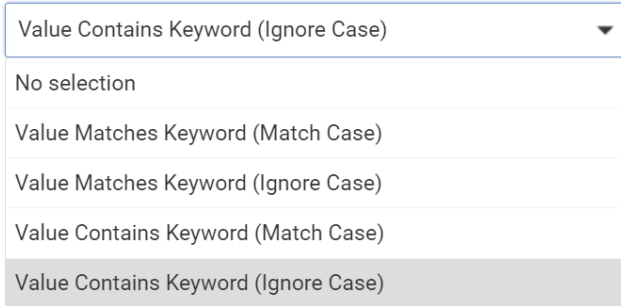
Tile Builder Grid: Advanced Topics

I am assuming that you have already been using a Tile Builder module in Advanced mode and are already familiar with how to use the features under the Highlights tab. Namely, Keywords, Thresholds and Format Rules (MAM) so in this section I will only point out a few changes.

Highlights Tab

Keywords**:** You can now select the type of match you wish to test for with the options shown below.

Keyword Match Type

Value Contains Keyword (Ignore Case) ▼
No selection
Value Matches Keyword (Match Case)
Value Matches Keyword (Ignore Case)
Value Contains Keyword (Match Case)
Value Contains Keyword (Ignore Case)

Thresholds**: No changes**

Format Rules**:** These work the same as they do in Multi-Attribute Monitor but there are additional examples that can be found in the "**Highlighting Help"** as shown below.

Format Rules: These are only available in Multi-Attribute Monitor & Grid, and are used to apply custom formatting to particular rows of the table. Common examples are:
- Progress Bar Example: %value%%[br][progress value=%value% max=100][/progress]
- Meter Example: %value%%[br][meter low=50 high=80 max=100 optimum=100 value=%value%][/meter]
- Direction Example: [style].dir{transform:rotate(%value%deg);font-size:38px}[/style][div class=dir]↑[/div] (%value%°)
- Speed Example: [style]@keyframes spin{0%{transform:rotate(0deg)}100%{transform:rotate(360deg)}} .sp1{animation:spin calc(5s / %value%) linear infinite}[/style][div class=sp1] ◎ [/div]
- Size Example: [span style="font-size:48px"]%time%[/span]
- Color Example 1: [span style=color:blue]%value%[/span]
- Color Example 2: [span style=color:%value%]%value%[/span]
- Background Color Example: [span style=background:orange]%sunset%[/span]
- Background Gradient Example: [span style='background:linear-gradient(to bottom, brown,orange);border-radius:30px;padding:3px'] ☀ [/span]
- Opacity Example: [span style="opacity:0.5"]%value%[/span]
- Tooltip Example: [span title='Last Event: %lastEvent% (%lastEventValue%) @ %lastActivity%']%deviceLabel%[/span]
- Marquee Example: [marquee]Last Event: %lastEvent% (%lastEventValue%) @ %lastActivity% %deviceLabel%[/marquee]

Replace Chars**:** This was introduced in Multi-Attribute Monitor and works the same way in Grid.

Cleanups

Cleanups are a way of modifying\formatting the data to be in a more pleasing form. A list of cleanups and their purpose can be found within the app under **Cleanup and Rule Help**.

With the added focus on event information in Grid 2.0 many cleanups have been added when using **Free Form mode** to perform date\time conversions and can be applied individually to each variable. Most of these are self-explanatory.

▼ Cleanup and Rules Help

▼ Cleanups

Cleanups are a way of modifying\formatting the data to be in a more pleasing form. Not all cleanups are available in all Tile Builder Modules

- None: The default value of no processing.
- Capitalize: Capitalize the first letter of the variable. Example: 'true' -> 'True', 'porch light' -> 'Porch light'
- Capitalize All: Capitalize the first letter of each word in the variable. Example: 'true' -> 'True', 'porch light' -> 'Porch Light'
- Commas: Adds commas to numeric values where appropriate. Example: '1842' -> '1,842'
- O Decimal Places: Rounds floating point numbers to nearest Integer (no decimal places). Example: '69.24' -> '69', '70.56' -> '71'
- 1 Decimal Places: Rounds floating point numbers to a single decimal place. Example: '69.24' -> '69.2', '70.56' -> '70.6'
- Upper Case: Converts all lower case letters to their upper-case equivalent. Example: 'true' -> 'TRUE', 'porch light' -> 'PORCH LIGHT'
- OW Code to Emoji: Converts an OpenWeather weather code to the nearest emoji equivalent. Example: weatherIcons attribute is '04d' - > '☁'
- OW Code to PNG: Converts an OpenWeather weather icon to a URL pointing to the PNG file located at OpenWeather.com
- Image URL: Wraps an Image URL within the correct HTML structure to display the image as embedded within the file.
- Remove Tags: Strips any HTML tags from the data.
- To: HH:mm - Converts a valid time to the form 19:35
- To: h:mm a - Converts a valid time to the form 7:35 PM
- To: HH:mm:ss - Converts a valid time to the form 19:35:26
- To: h:mm:ss a - Converts a valid time to the form 7:35:26 PM
- To: E HH:mm - Converts a valid time to the form Tue 19:35
- To: E hh:mm a - Converts a valid time to the form Tue 19:35 PM
- To: EEEE hh:mm - Converts a valid time to the form Tuesday 19:35
- To: EEEE hh:mm a - Converts a valid time to the form Tuesday 19:35 PM
- To: MM-dd HH:mm - Converts a valid time to the form 4-9 19:35
- To: MM-dd HH:mm a - Converts a valid time to the form 4-9 7:35 PM
- To: MMMM-dd HH:mm - Converts a valid time to the form April 09 19:35
- To: MMMM-dd HH:mm a - Converts a valid time to the form April 09 7:35 PM
- To: yyyy-MM-dd HH:mm - Converts a valid time to the form 2024-04-09 19:35
- To: yyyy-MM-dd HH:mm a- Converts a valid time to the form 2024-04-09 7:35PM
- To: dd-MM-yyyy h:mm a - Converts a valid time to the form 092024-04-09 7:35 PM
- To: MM-dd-yyyy h:mm a - Converts a valid time to the form 04-09-2024 7:35 PM
- To: E @ h:mm a - Converts a valid time to the form Tue @ 7:35 PM
- To: Elapsed Time (dd):hh:mm:ss - Converts a valid time to an elapsed time from that date in the form 5d 14h 41m 44s. Days are only displayed if 1 or greater.
- To: Elapsed Time (dd):hh:mm - Converts a valid time to an elapsed time from that date in the form 5d 14h 41m. Days are only displayed if 1 or greater.
- To: Remaining Time (dd):hh:mm:ss - Converts a valid time to a remaining time until the future date provided in the form 3d 9h 15m 52s. Days are only displayed if 1 or greater.
- To: Remaining Time (dd):hh:mm - Converts a valid time to a remaining time until the future date provided in the form 3d 9h 15m. Days are only displayed if 1 or greater.
- **ALL values are instantaneous and are ONLY calculated when the table is refreshed.**

The last four of the date\time cleanups for **Remaining Time** and **Elapsed Time** can be used to convert a date\time into a **Remaining Time** for a future date\time, or into an **Elapsed Time** for a past date\time for an event such as a planned filter change or the date\time a battery was replaced.

Runtime Tags

These are hidden HTML tags that are added to the value of any Duration Field. In the following example you will see the HTML tag [rt-] where rt stands for runtime and – (or +) indicates the current state of the device.

bathRoomFanRunTime: (56m 26s) ([hqq3] ([rt-]56m 26s) [/hqq3]),

In the above example the switch for this fan is currently off so the runtime is indicated as [rt-]. If the switch were currently on it would be indicated as [rt+]. This same rule holds true for all other uses of lastXXXDuration whether it be open/close, active/inactive etc.

The purpose behind these tags is to make it easy to highlight these values to indicate the present state of the device. Here is how you do that.

1) Configure the Rule – All Keywords for the variable that holds the lastOnDuration for example.

Source #14	Var Name	Device	Device Detail *	Cleanup	Rules
Device Detail ▼	bathRoomFanR	Bathroom Vent Fan	lastOnDuration ▼	None ▼	All Keywords ▼

2) Configure two keywords. One for rt- and one for rt+ and assign the text colors and size that you wish to use.

Keyword Match Type	Keyword #3	Replacement Text #3		Color (#ffffff)	Text Scale
Value Contains Keyword (Ignore Case) ▼	rt-	(%value%)			100 ▼

Keyword Match Type	Keyword #4	Replacement Text #4		Color (#000000)	Text Scale
Value Contains Keyword (Ignore Case) ▼	rt+	%value%			100 ▼

3) I also like to change the background color and you can do that easily by enabling overrides and entering a background color for Highlight 3 and 4 as those are used to format the output of keywords 3 and 4.

Settings Overrides
```
#high3#=background:red|#high4#=background:lime
```

With these updates my tile for tracking the automated fans in my house now looks like this. The **Circulation Fan** is not currently running but the last time it did run it was for 13 hours, 0 minutes and 4 seconds. The **Attic Fan** is on and has been running for 4 minutes and 46 seconds.

Automatic Fan Status @ 09:28 AM	
Circulation Fan (±8°) (Basement ↔ Bedroom)	67°F - OFF - 74°F (13h 0m 4s)
Attic Fan (±15°) (Attic ↔ Outside)	70°F - ON - 54°F 4m 46s
Bath Vent Fan (±10%) (Bathroom ↔ Living Room)	33% - OFF - 38% (56m 26s)

Refreshing Timer Data (Free Form Mode Only)

Tables will refresh automatically whenever one of the monitored attributes changes. In my laundry monitor shown below the tile also monitors the Washer and Dryer power information which is refreshed every 30 seconds. Whenever that information refreshes, all the time related data is also refreshed, so it works well without any extra accommodations.
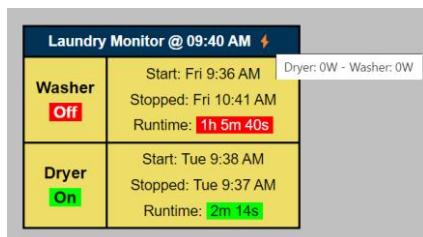
However, if I did not have power monitoring plugs for my washer and dryer the table would only update when the washer of dryer received On or Off events. Not very helpful if I want to see how long the Dryer has been running. But it is relatively easy to ensure that we get updates whenever we want without doing continuous refreshes. Here is how you can accomplish that.

1) Add a virtual device called **Dashboard Refresh** (or something similar). A simple switch is all that you need but I found a momentary device to be preferable.

2) Add **Dashboard Refresh** as monitored attribute to your Tile Builder configuration. It does NOT need to be placed onto the table itself.

3) Check your publishing settings. If you want to get immediate updates you must configure you publishing settings like this to eliminate any delays.

4) Add the **Dashboard Refresh** device to the actual dashboard.

5) **Viola!** Clicking on the device causes a change of device state which causes **Tile Builder** to regenerate the table with the most current information.

| | For myself I chose to add a menu bar with a refresh option which uses the Maker API to do a silent button press. You must replace the various parameters with your own values.

[iframe name=a width=0 height=0 frameborder=0][/iframe][a href=http://192.168.0.200/apps/api/3691/devices/4284/push?access_token=511c1795-XXXX-XXXX-ea49218c6cbe target=a]Refresh[/a] |

Open Weather Customizations

Open Weather seems to be the most prolific weather app used by the Hubitat community, so I added a couple of "**Cleanups**" related to Open Weather.

Load Image

The first one we used earlier which forced a string URL to load as an image. Technically this is not strictly Open Weather specific, but it is the only source for which I have tested compatibility.

Use Weather Emoji (OW Code to Emoji)

Open Weather uses a code within the **weatherIcons** attribute that can be used to map the current weather to the closest available weather eMoji. This can be used where saving space, both in terms of footprint and byte size is a primary concern. See example below.

Use Open Weather Icon (OW Code to PNG)

This also uses the **weatherIcons** attribute that can be used to map the current weather to the designated Open Weather icon as shown below.

Weather Classes

There are a couple of classes listed under the **Text Fields and Variables Help\Advanced HTML Examples** that may be useful in displaying a weather tile and are not specific to Open Weather.

- Direction Example: [style].dir{transform:rotate(%value%deg);font-size:38px}[/style][div class=dir]↑[/div] (%value%°)
- Speed Example: [style]@keyframes spin{0%{transform:rotate(0deg)}100%{transform:rotate(360deg)}} .sp1{animation:spin calc(5s / %value%) linear infinite}[/style][div class=sp1] ◎ [/div]

Viewing Variable HTML

At times it is helpful to be able to view the underlying HTML that might be affecting a variable. You can do exactly that by changing the **Show Variables** control to **Show Variables & HTML** as shown.

Show Variables?

Show Variables & HTML ▼

In the example below I have added a rule to mark the **feelsLike** temperature variable in blue when the temperature falls below 35°F.

feelslike: 33 ([hqq6]33[/hqq6]), forecastHigh: 38 (38), forecastHigh1: 33 (33), forecastLow: 31 (31), forecastLow1: 28 (28), icon: ([img src=https://tinyurl.com/icnqz/34.png]), icon1: ([img

src=https://tinyurl.com/icnqz/34.png]), weatherEmoji: ([img src=https://openweathermap.org/img/wn/02d.png]), windspeed: 5 (5),

Publishing

Publishing your tile to the dashboard is just the same as in other Tile Builder modules.

Full Screen Mode

Tile Builder Grid can have a lot of controls in use when creating a complex tile. To make the best use of the screen Tile Builder has always had collapsible sections using the headers. In Grid I have introduced a "Full-Screen" mode. This simply hides some of the Hubitat surrounding elements to leave more room for the important stuff as shown below.

Normal
Hubitat Wide (Double arrows by the gear)
Full Screen (Arrow by Tile Builder Grid title text)
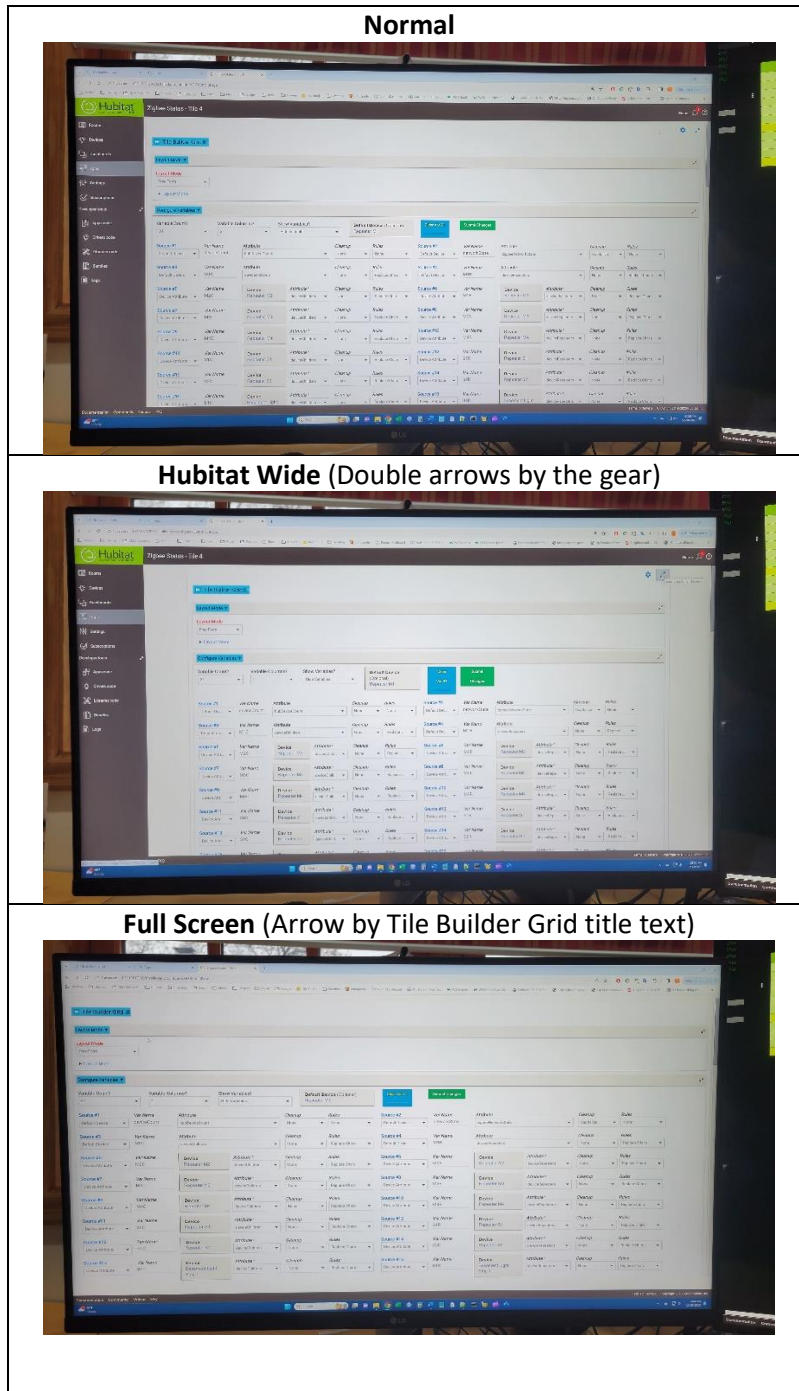
Using Font-Awesome Icons

Using Overrides you can load alternate fonts and Icons if desired. Go to the Advanced Tab and click on Enable Overrides:

1) Add the following text to your Overrides area: ***#head#=[link rel=stylesheet href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.css']***

2) Using a URL shortener I get this *http://tinyurl.com/5956hskv so I could use: **#head#=[link rel=stylesheet href='http://tinyurl.com/5956hskv']** which saves some space.*

If you have other commands in Overrides already you will need to separate them with the vertical bar symbol like this command1 | command 2 | command3

You can now place a Font-Awesome Icon using the following syntax in any text field: ***[i class='fa-solid fa-temperature-full']*** *Doing so will render this icon* 🌡 *in whatever your selected Font Size is.*

You can find a searchable list of free Font-Awesome icons at this URL: ***https://fontawesome.com/search?o=r&m=free***

When you click on an Icon you will get a popup like this:

Clicking on the code snippet will place it in the copy\paste buffer and you can paste it into any Tile Builder text field like this:

You are not quite finished! You must change any <> to [] and any " to ' so the final result looks like this.

Note: Providing a closing tag is optional if it is the last item in the field so the [/i] could be omitted.

Hub Info Example

Here is how I built my Hub Info Tile.

Configure Variables (Partial list)

Configure Variables ▼					
Variable Count?	Variable Columns?	Show Variables?	Default Device (Optional) Hub Info	Clear Var 16	Submit Changes
16	1	Show Variables			

Source #	Var Name	Attribute	Cleanup	Rules
Source #1 — Default Device	dbSize	dbSize	0 Decimal Pla...	None
Source #2 — Default Device	lastRestart	lastHubRestartFormatted	None	None
Source #3 — Default Device	hubUpdateSta	hubUpdateStatus	None	All Keywo...
Source #4 — Default Device	cpu5	cpuPct	None	None
Source #5 — Default Device	cpu15	cpu15Pct	None	None
Source #6 — Default Device	zigbeeStatus	zigbeeStatus	Capitalize	None
Source #7 — Default Device	zWaveStatus	zwaveStatus	Capitalize	None
Source #8 — Default Device	connectType	connectType	None	None
Source #9 — Default Device	freeMem	freeMemory	Commas	None

Grid Layout

Grid Layout ▼		
How Many Data Rows?	How Many Data Columns?	Auto Refresh Table on Save?
9	2	True

R1C1
[i class='fa-solid fa-temperature-full']

R1C2
%hubTemp%°F

R2C1
[i class='fa-solid fa-memory']

R2C2
[z title='5 / 15 min averages']%freemem% / %freemem15%

R3C1
[i class='fa-solid fa-microchip']

R3C2
[z title='5 / 15 min averages']%cpu5%% / %cpu15%%

R4C1
[i class='fa-solid fa-power-off']

R4C2
%lastRestart%

R5C1
[i class='fa-solid fa-network-wired']

R5C2
%connectType%

R6C1
[i class='fa-solid fa-database']

R6C2
%dbSize% Mb

R7C1
[i class='fa-solid fa-code']

R7C2
%version%

R8C1
[i class='fa-solid fa-cloud-arrow-down']

R8C2
%hubUpdateStatus%

R9C1
[i class='fa-solid fa-hockey-puck']

R9C2
%hubModel%

Result

Home Hub @ 10:50 AM	
🌡	104°F
🏛	509,908 / 514,093
⚙	2.0% / 2.08%
⏻	01/01/2024 @ 04:14PM
🖧	Ethernet
🗄	6 Mb
</>	2.3.6.146
☁	Update Available
🗄	C-5

Alerts: platformUpdateAvailable
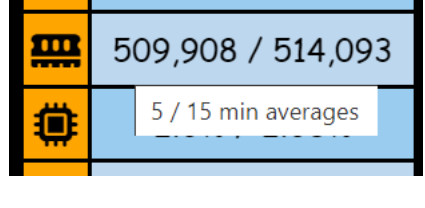
To add the orange background to the first column and get the Icons the size I wanted I added this to my overrides: ***#class1#=td:nth-child(1) {font-size:20px !important; background:orange !important}***

This says for the 1st column make the font size 20 pixels with an Orange background. The !important tells HTML\CSS that this setting should override all competing settings.

Adding a Title/Tooltip

The ***[z title='5 / 15 min averages']%freemem% / %freemem15%*** *uses the HTML Title command to add a ToolTip to the text which I used for fields* that have two numbers to identify the significance of each. The letter z is just an assigned class name and has no relevance, other than being very short.	509,908 / 514,093 5 / 15 min averages

Overrides now looks like this:
Settings Overrides ↔
`#head#=[link rel=stylesheet href='http://tinyurl.com/5956hskv'] |`
`#class1#=td:nth-child(1) {font-size:20px !important;background:orange !important}`	

Highlighting Updates

Whenever there is an update, it will be shown in green text. This is done using a **Keyword** which can be found under the **Highlights** tab.

Assign Keyword testing to the Variable.	Source #3 Var Name Attribute Cleanup Rules Default Dev... hubUpdateSta hubUpdateStatus None All Keywo...
If the result contains "**Update Available**", make it green**.**	Highlights How Many Keywords? Show Keywords ▼ 1 Keyword Match Type Enter Keyword #1 Replacement Text #1 Highlight 1 Color (#008000) Highlight 1 Text Scale Value Matches Keyword (Ignore Case) Update Available [b]%value%[/b] 100

It may seem a little tricky at first, but you will get used to it quickly.