

SmartThings Device Handler Notes

Tasmota for RGBCCT Bulb w/MQTT for ST Classic V1.1

This is the quick guide for the SmartThings device handler: "Tasmota RGBCCT Bulb w/MQTT for ST Classic V1.1"

Tasmota firmware has a rich set of capabilities that are not being fully exploited by the currently available device handlers (DH) for Tasmota based RGBW devices. This DH makes a large range of commonly used capabilities available through the SmartThings Classic interface. It supports the following features:

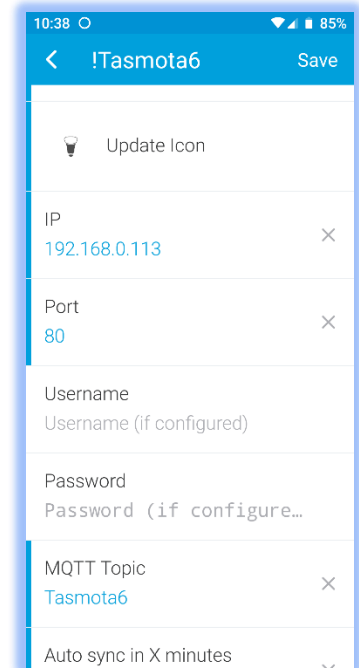
Capability	Direct IP (1:1)	MQTT (1:N)
Off\On	Yes	Yes
Dimmer	Yes	Yes
Color	Yes	Yes
Color Temperature	Yes	Yes
3 x Preset Whites	Yes	Yes
3 x Configurable Colors	Yes	Yes
Sync from Bulb	Yes	Yes*
Fade (adjustable)	Yes	Yes
Boot Color	Yes	No
Poll\Sync	Yes	No
Update DNI	Yes	No
WiFi Strength	Yes	No

*The Device Handler will sync from the device with the specified IP address. It is expected that the device is subscribed to the MQTT Group Topic (if used) so it is a partial reflection of the state of an MQTT bulb group.

Device Setup

After installation of the DH and the creation of a device you must configure the IP address of the device and the MQTT Group Topic (optional). All other values have reasonable defaults.

After saving the configuration, the Device Network Interface (DNI) will be updated automatically and can be seen at the very bottom of the main screen. It should be in the form, DNI:XXXXXXXX:0050 where the X's are hex values.



If this does not update correctly, then neither IP nor MQTT communication will work. Clicking on the DNI button will force the DNI to be re-calculated and applied.

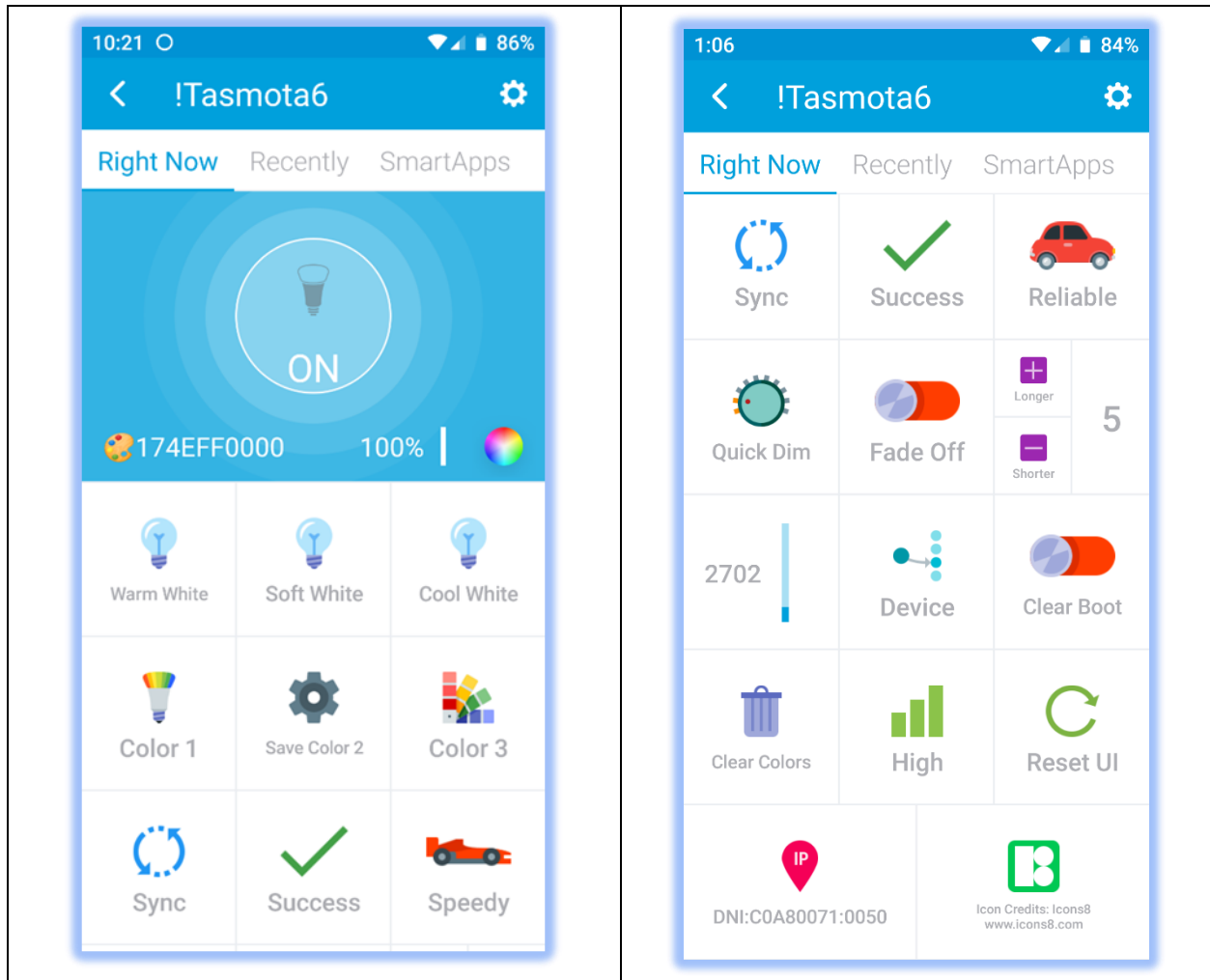
Once the DNI is set properly you can now sync the bulb to the UI using the Sync tile. If you see a "System Busy" message it typically means that the background polling process is already running. If it stays this way for more than 15 seconds click the Reset UI to clear it.

MQTT Setup Notes

If you are dealing with a single bulb then you don't need to configure an MQTT Topic. It really adds nothing to a single device scenario.

However, if you have multiple bulbs in a room the DH can control the entire group of bulbs simultaneously using MQTT. All you must do is enter the MQTT group topic to which they are subscribed, for example "BedroomLights" and then toggle from Device to MQTT on the main screen. All commands will then be sent as Tasmota MQTT publish commands to the specified MQTT topic via the bulb associated with the IP address.

Quick Guide



Most of the controls are self-explanatory or will become obvious with a little playing but here is the quick guide.

On\Off Button: Self-explanatory.

Color Wheel: Select a color to apply to the bulb(s). Selected colors can be saved to one of the available Color1...3 tiles.

Dimmer: Set the dimmer range anywhere from 0 to 100. Note that dimmer settings are preserved across different white selections. Selecting an RGB color will automatically adjust the dimmer to 100. This action is performed by Tasmota, not the DH.

Message: Will typically show the currently selected color in hex but will also show other system messages.

Color Temperature Row: These use the preset Kelvin values in the settings to quickly select a common white. These presets can be changed in Settings if desired but the labels remain the same.

Color Row: These tiles have three states. The "Save Color 2" tile shows that it presently does not have a color associated with it and is available to save a new color. The "Save Color 3" tile shows that it does have a color associated with it. Clicking on that tile will set the color of the bulb to the saved Color 2. The "Save Color 1" tile shows that it has a saved color and is presently selected. These can be reset using the "**Clear Colors**" tile.

Sync: Forces a sync between the UI and the Bulb. The UI settings will be changed to match those of the bulb. This tile will also show as active when the background poll process is active (every 15 mins by default).

Status (Success): This is a status tile that shows the status of a request. The status will be one of the following; idle, send, wait, receive, success or fail. A request is not complete until the status reads either Fail or Success.

Speedy\Reliable: This tile has two states which affect the underlying communication. In Reliable mode all requested changes are considered transactions which are not complete until a response is received that matches the requested value. For example, if we set the Dimmer to 50 then we must have positive confirmation back from the device that the Dimmer value is now 50.

In Speedy mode Off\On, Color, Color Temperature, Dimmer and Fade Speed are not transacted. In Speedy mode the DH does not wait for a response from the bulb and just assumes that the command has gone through. This makes the UI faster but introduces the possibility of the UI and the bulb no longer being in Sync. I'd recommend using this mode unless you find it to be unreliable.

QuickDim: Toggles the dimmer between two preset values of 20% and 50%. These values can be changed by going into settings.

Fade Group: There are 4 tiles in this group. Fade Off\On, Longer, Shorter, FadeSpeed (default 5). Tasmota uses a fade speed range from 1 – 20. Higher numbers correspond with a longer fade transition period. Fade applies whenever there is a color change, or a soft power on\off occurs. Once Fade is applied the Fade tile changes to "Fade Off" and can be removed by a second press of this button.

Color Temperature: This untitled vertical bar allows the setting of a CT between 2000 and 6535 Kelvin. Tasmota CT is configured in Mireds and supports a range between 153 and 500. $\text{Kelvin} = 1,000,000/\text{Mireds}$.

Device\MQTT: This tile has two states, Device and MQTT. When Device is selected the communication is 1:1 and only one device is affected (the one specified by the IP address). When MQTT is selected the DH sends all commands to the MQTT Topic using Tasmota's MQTT publish capabilities. This is very useful when dealing with a group of bulbs as they can all be changed simultaneously if they share an MQTT topic.

When using MQTT mode Tasmota does not respond with the device status, it only acknowledges that the command was received. The DH assumes that if a response is received, the operation was a success. As such it cannot tell if only 2 out of 3 bulbs came on. When in MQTT mode the sync operation is performed against the bulb that corresponds with the IP address. Naturally for MQTT mode you must have a functional MQTT network.

Boot Color: This option is only available in Device mode. When selected it applies\overwrites a rule to Rule3 which forces the bulb to the currently selected color when there is a hard power on. This is useful in ensuring that a bulb always returns to a certain color at boot up, regardless of any prior changes. Boot Color is a toggle and once Boot Color is applied the toggle changes to "Clear Boot". Once the Boot Color is cleared the bulb will come back on with whatever color was applied when the power was turned off.

Clear Colors: If you wish to change the values of Color1...Color3 this button will wipe out any saved values and allow you to start over.

WiFi Icon: This tile merely shows the signal strength of the Bulb associated with the configured IP address. A bulb operating with a "Low" Wifi status is more likely to have communication issues.

Reset UI: In the event of the UI becoming out of sync or a system busy message lasting more than 15 seconds this tile performs a reset and terminates the current command.

DNI:XXXXXXXX:0050 This is the Device Network Address that is comprised of the IP address in hex format and the HTTP port in hex format (0050 is port 80). This should be calculated and applied when the IP address is assigned and preferences saved. If the DNI tile shows the name you originally assigned it when creating the device, then there is something wrong. Pressing the DNI tile will cause the DNI to be recalculated and applied if it is incorrect. The only time this process is problematic is if you

are creating and deleting multiple devices in short order that use the same IP address. Check the log and check that no other device is already assigned that IP address.

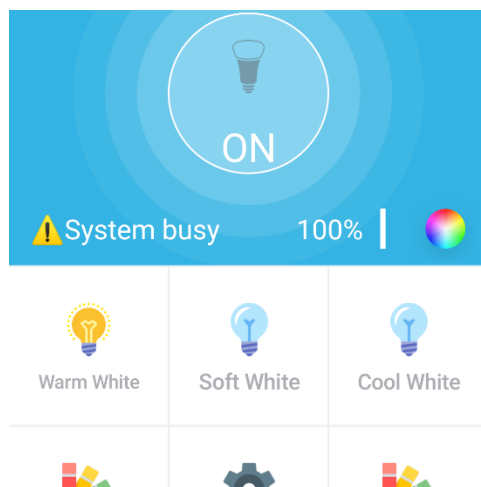
Icon Credits: This tile does nothing other than acknowledge that all tile icons used in this DH are courtesy of Icons8 (www.icons8.com).

Backgrounder

SmartThings HTTP calls are asynchronous which makes for a challenging paradigm. It is not uncommon for an HTTP request to fail or timeout somewhere within the SmartThings platform, in which case the device handler never receives a response to process. A tricky problem because the programmer cannot distinguish between a request time out (typically device off) and a response time out (typically network issue). Two very different scenarios, leaving the device in different states, with no obvious distinction.

In this DH I have turned these stateless calls into a quasi-synchronous transacted model by waiting for a response to an HTTP call and correlating the returned state to the requested state.

Because of this synchronous mode we wait (in Reliable mode) until we get a response or a response times out before we permanently commit the changes to the UI. Because of this transacted model it is only possible to execute one command at a time. The DH will block any additional commands until the current command has cleared. This will be indicated on screen as shown below with a System Busy message. When the current command is finished the status will clear.



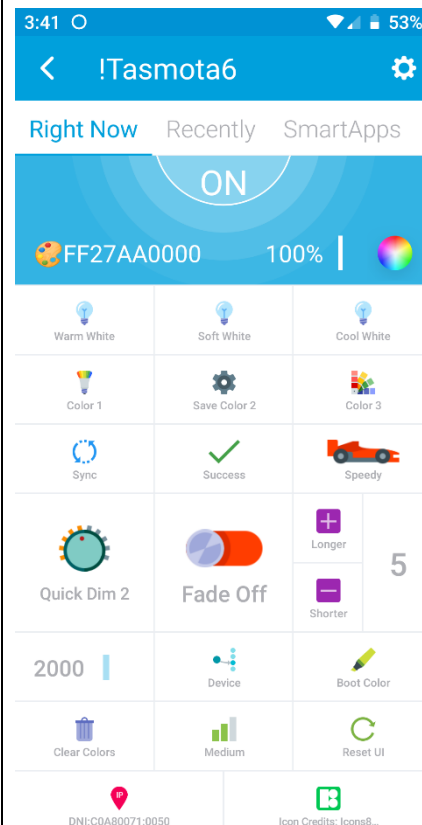
Tips: There are few user modifiable settings within the code that you can change without any knowledge of Groovy.

Dense Mode: If you have good near vision and prefer a denser display you can change the density very easily.

Locate the function `mintileheight()` and modify the return value.

```
32 //*****
33 // Start of user modifiable functions
34 private mintileheight(){
35     //Normal mode return 2
36     //Dense mode return 1
37     return 1
38 }
39
```

The DH display will now look like this.



Logging: You can easily tweak the amount of logging that goes on by changing the log threshold. The normal level of logging is zero, increasing it to 1 or 2 will increase the amount of logging.

```
//Function to selectively log activity based on various logging levels. Nor
//Loglevels are cumulative: -1 All errors, 0 = All user actions plus statu
private log(name, message, loglevel){

    //This is a quick way to filter out messages based on loglevel
    def threshold = 0
    if (loglevel > threshold) {return}

    //This is a quick way to filter out messages from a given function
    def filterlist = "listofnamestofilter: function1, anothername"
    if (filterlist.contains(name) == true) {return}
```

Troubleshooting

I have spent a lot of time writing and testing this DH but as with most first releases of software there will be some bugs. I'd be happy to fix them, but I need to be able to reproduce them in order to do so. If you are responding on GitHub or one of the forums please be as specific as possible with any issues. If they are reproduceable, please document the steps required to do so.

If you have some Groovy knowledge the first thing to do is to look at the SmartThings Logging for the device for any exception errors. You can temporarily turn up the logging as described earlier. Any null values would be of special interest.

You can also go to the Tasmota console on the device that should be receiving the commands and see if they have been appeared and been executed without error.

If you are having issues with MQTT make sure your group of bulbs are all registered to the same MQTT group topic. You can do this by going to the Tasmota console on each of the devices and issuing the command "GroupTopic <name>". Replace <name> with something that makes sense such as "BedroomLights". Remember that MQTT Topics are case sensitive.

Now test it out from your MQTT console. If you are using Mosquitto this command would turn on all the lights in the group.

`mosquitto_pub -t cmnd/BedroomLights/power -m on`

If this works as you expect, you can now try it with the DH in MQTT mode.

Written by: Gary Milne on December 4th, 2019.