HCA Tech Note 613

Hubitat Integration (updated 8-July-22)

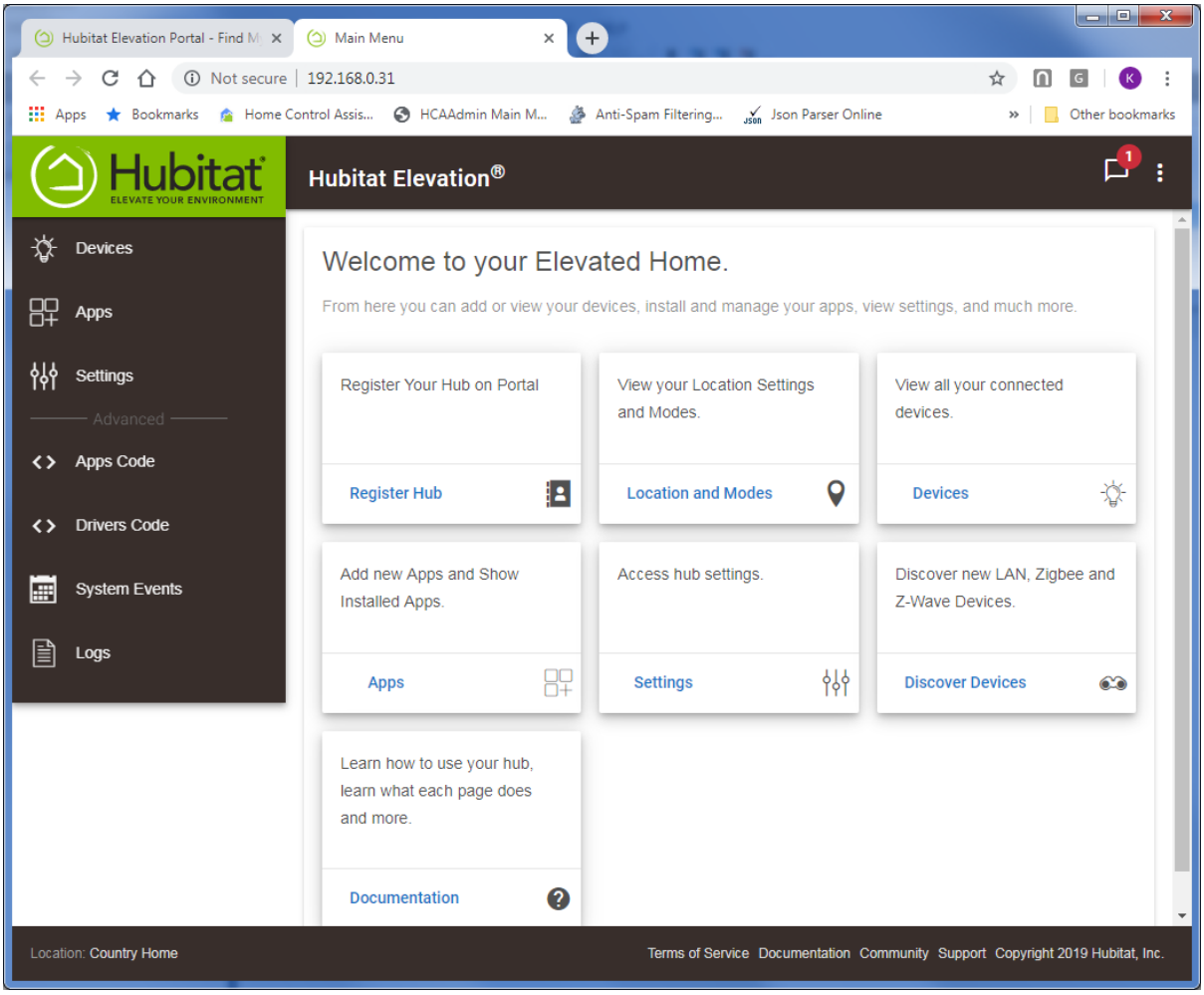
Hubitat is an integrator of various protocols that uses a hub that provides a protocol independent interface to devices of different types – a bit like HCA. The Hubitat app allows you to add, view, and modify devices, and then to perform simple actions on them. The advantage of having HCA support Hubitat is that all the device types they support are then available to HCA.

For HCA support, a use must have a Hubitat hub installed. Any devices that will be controlled by Hubitat need to be added to their system so the hub can control and receive signals from them.

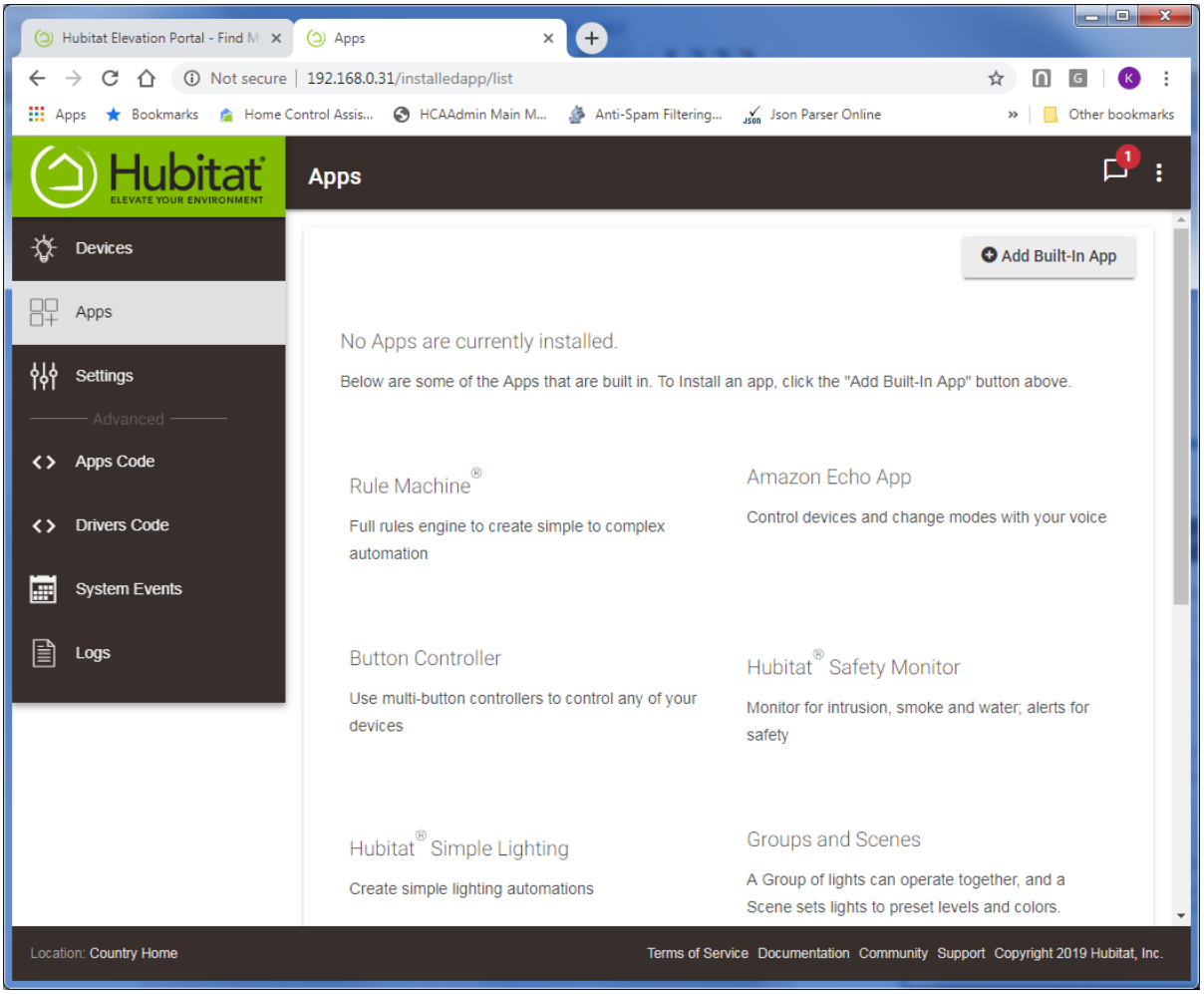
NOTE: Hubitat works in two ways: either local control or control via the Hubitat Cloud. The HCA implementation uses Hubitat only in local mode. This requires that you determine the IP address of the Hubitat hub and ensure it doesn't change. Investigate what facilities your network router has for maintaining the same IP address for a device over time. Some routers call this "IP Reservations" or provides some facility in the LAN setup section. Each router is different but most modern ones have this ability someplace in the settings. To receive events from the Hubitat hub into HCA, the IP address of the computer running the HCA Server must also not change.

Follow the instructions in the Hubitat documentation to setup your hub. In the portal.hubitat.com website there is a "Find Hubs" button. This locates your hub IP address.

HCA Tech Note 613

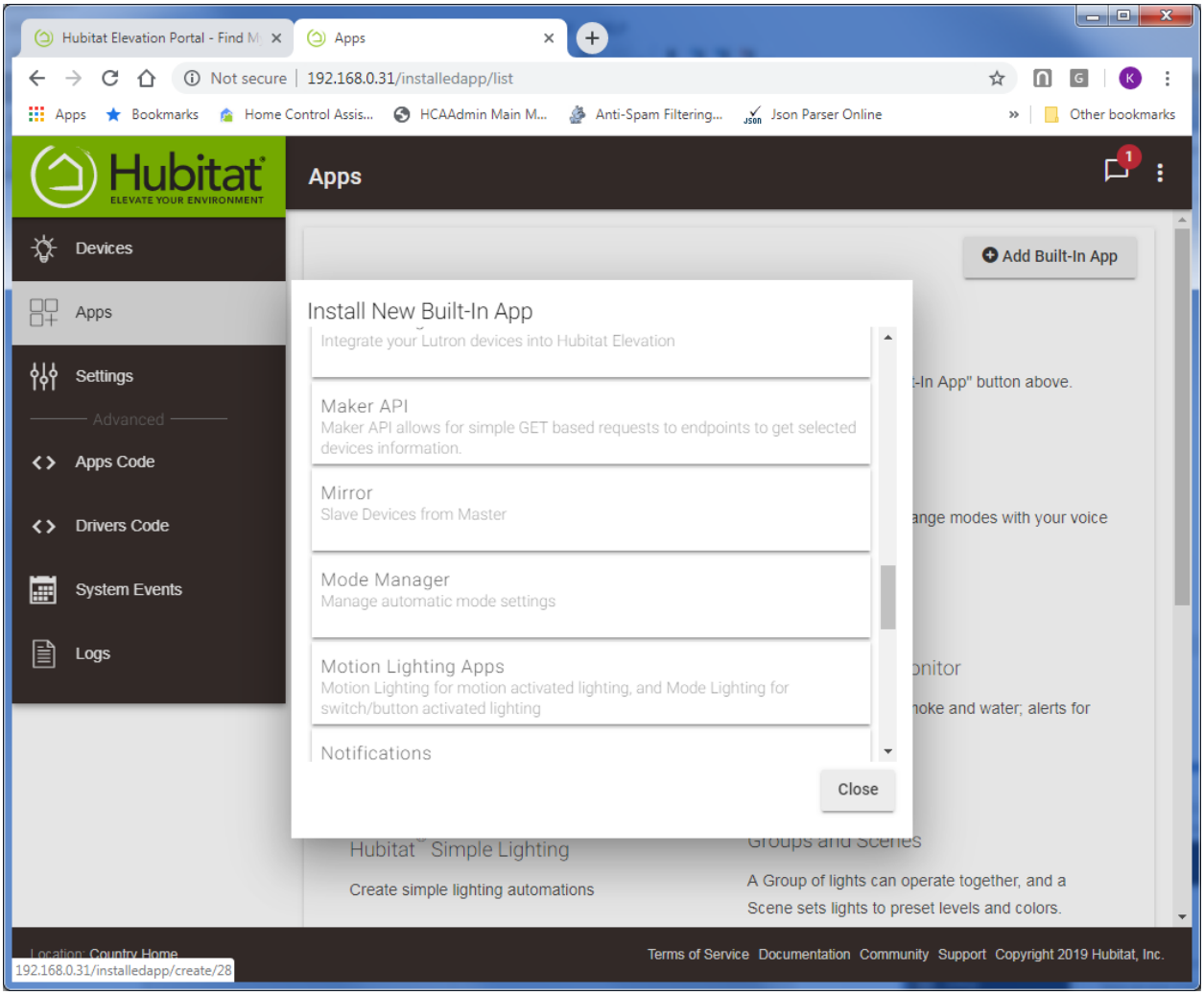


Take note of the IP address shown in your browser. Next, click on “Apps” in the left menu.

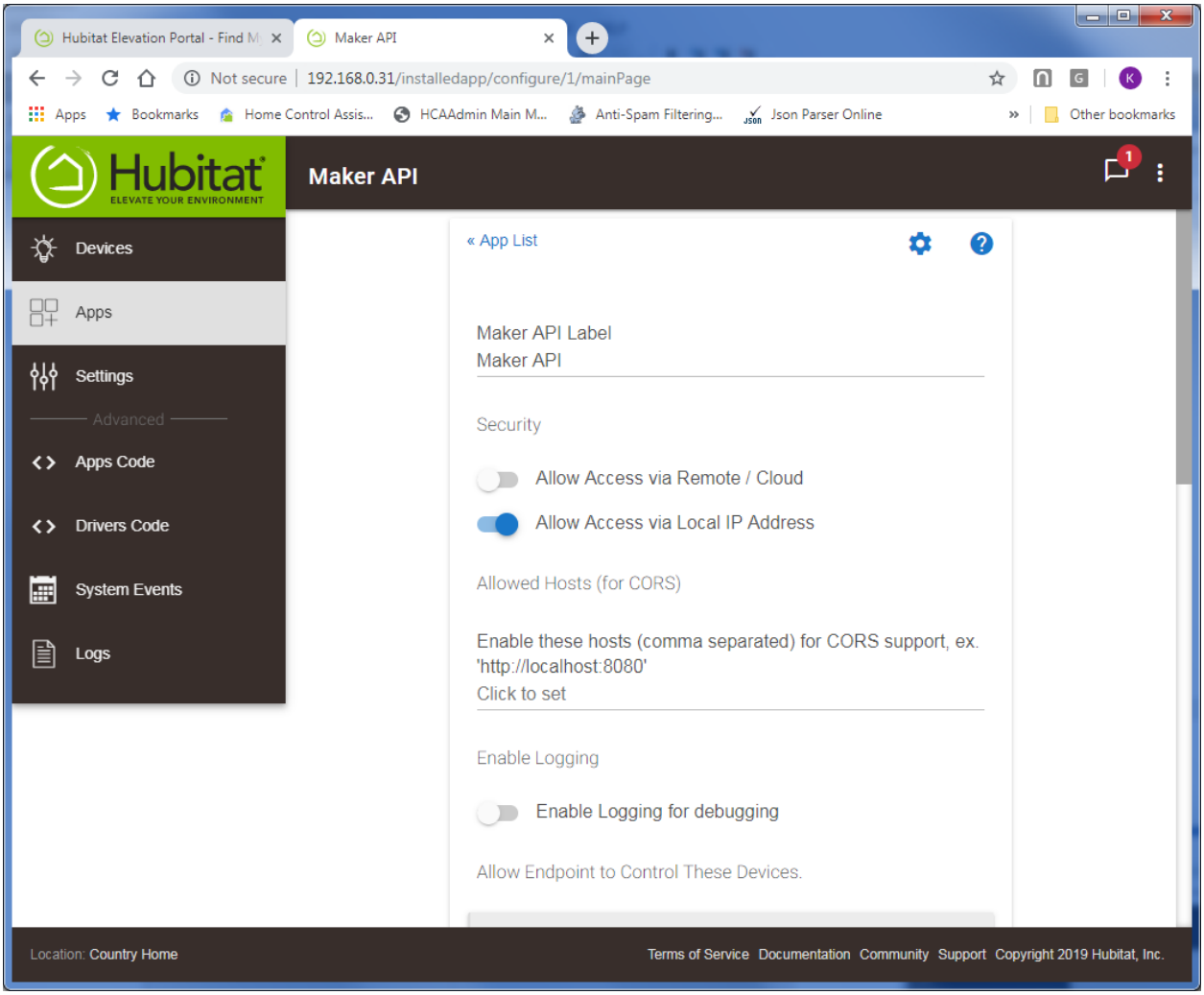


Press the “Add Built-in App” button.

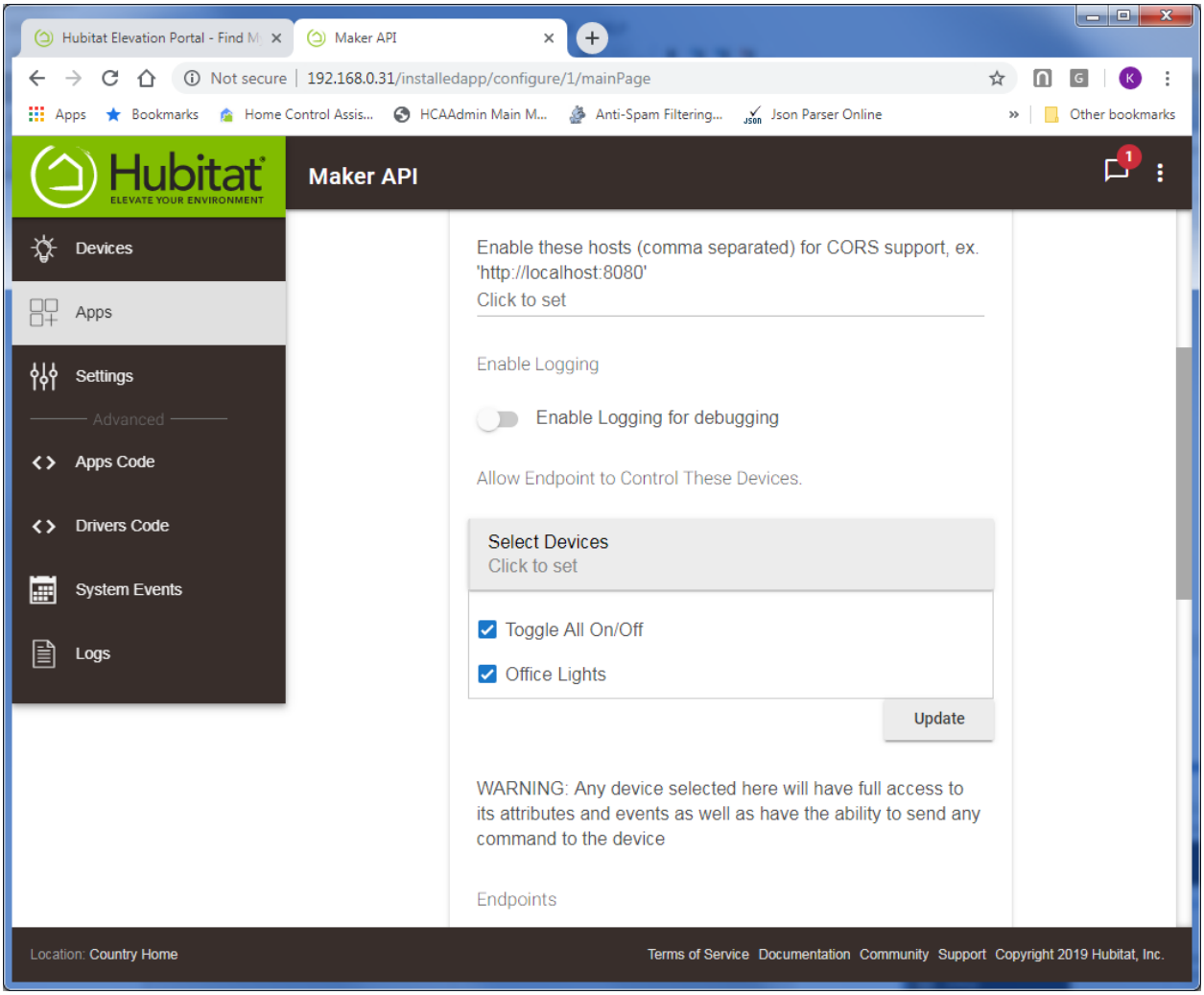
HCA Tech Note 613



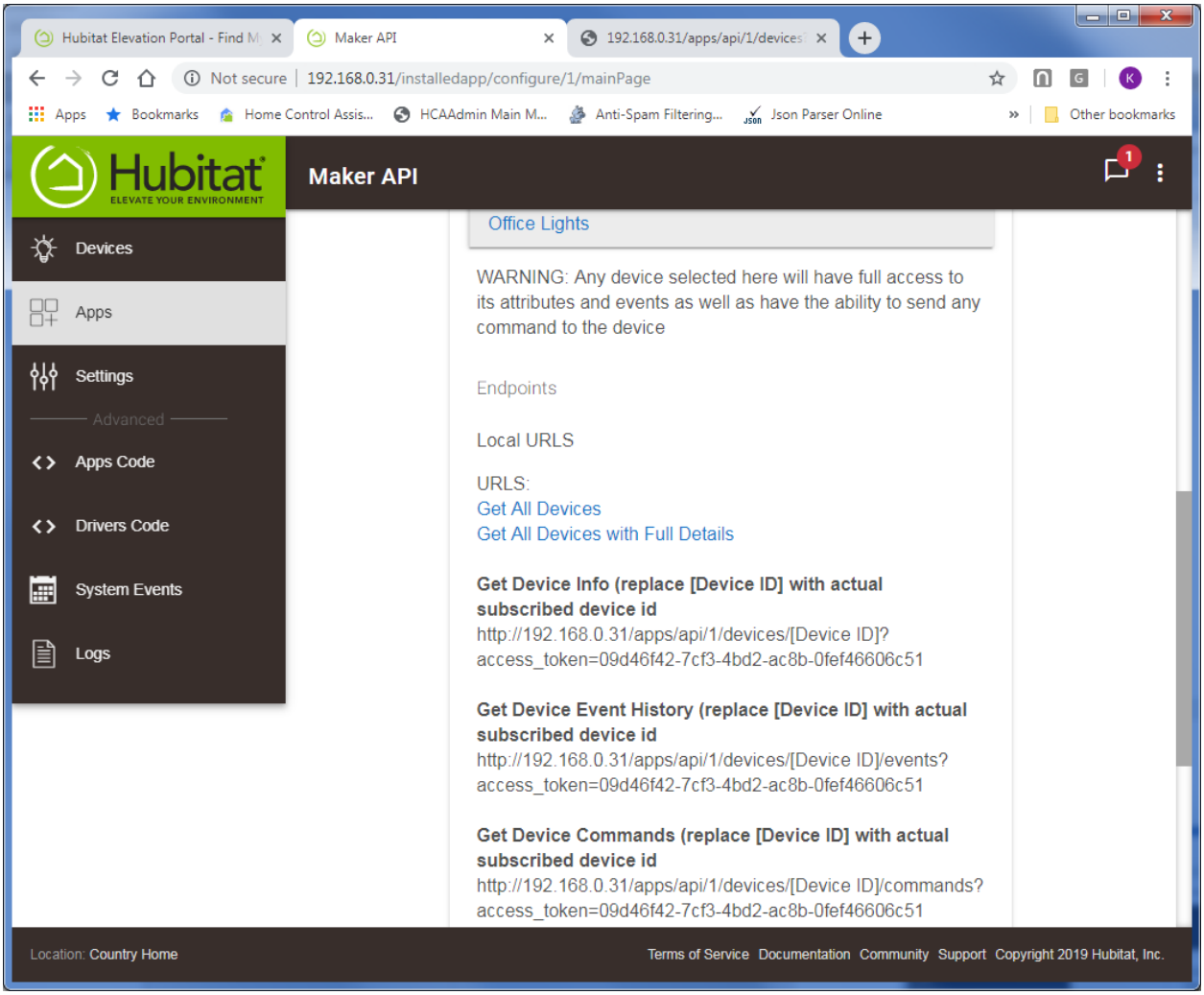
Select the “Maker API”



Slide the option “Allow Access via Local IP Address” to the right to enable it.



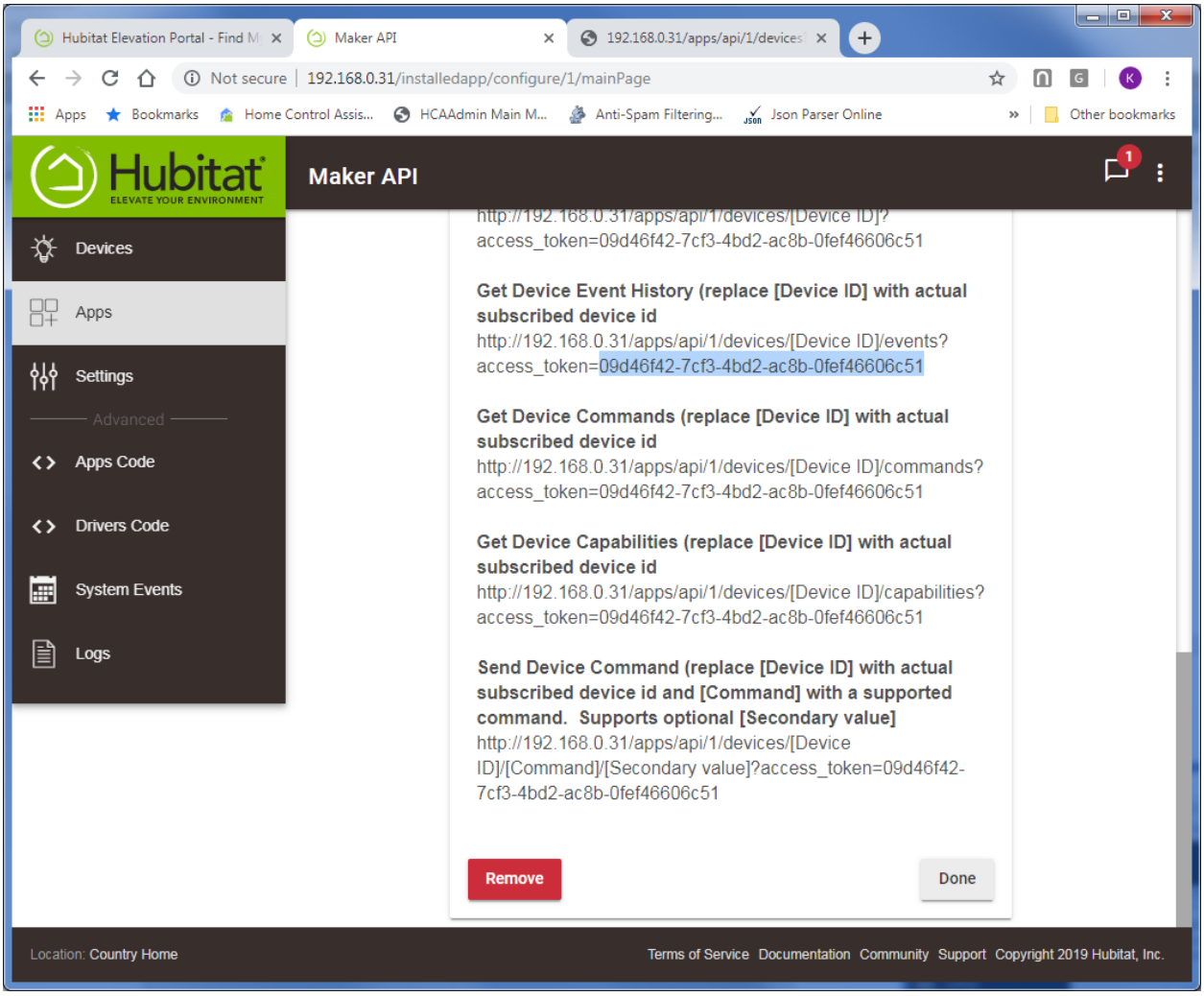
Scroll down and tick the checkbox next to the device you want to allow HCA to control.



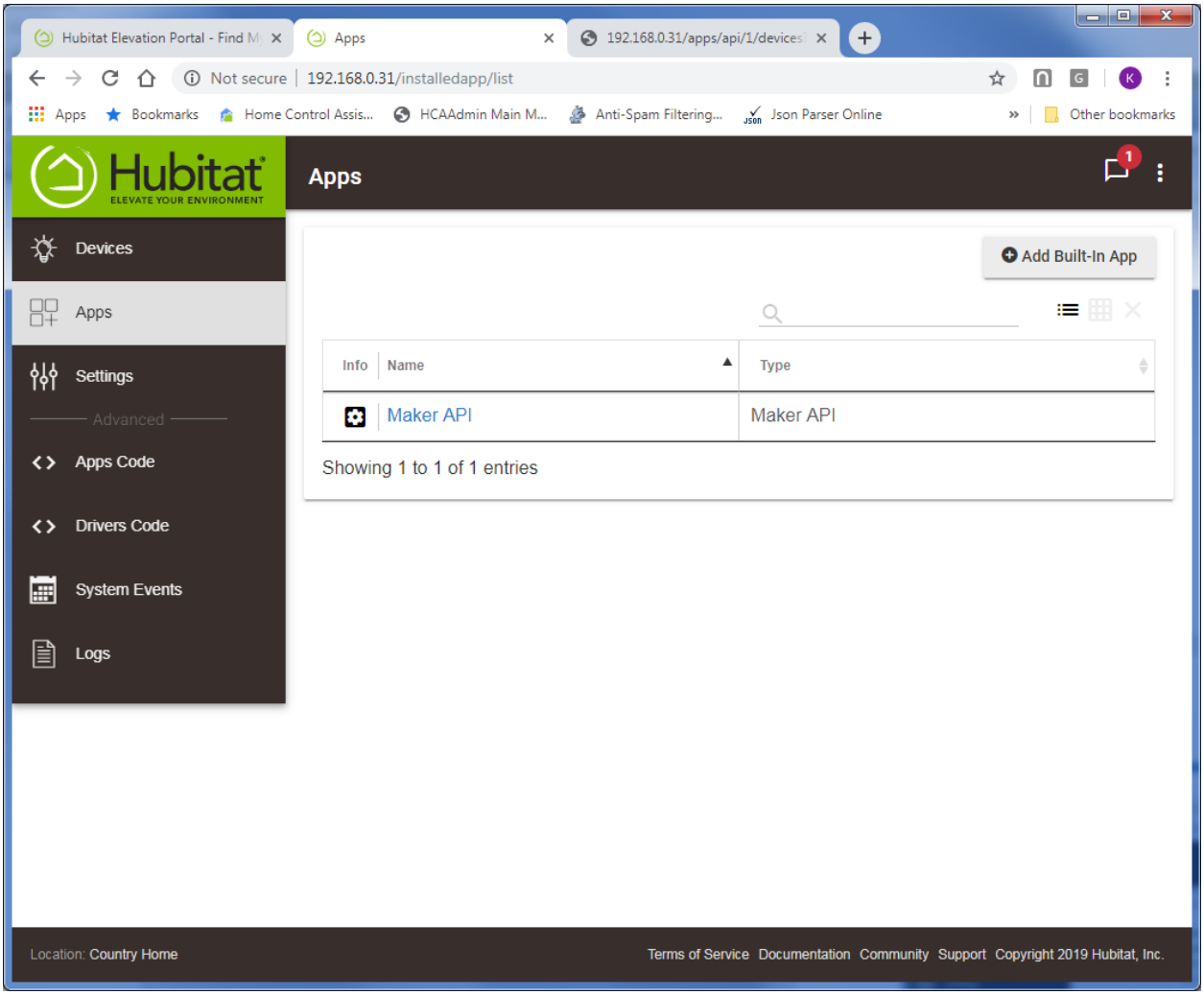
Again, scroll down and take note of two key pieces of info:

1. Look at any of the URLs it displays and extract the number between the "/api/" and the "/devices". In the above example that is "1" but yours may be different. Later on you will save this in a HCA global variable called HE_APICode
2. Extract the text from the end of the URL following the "access_token=". In the above example, it is: 09d46f42-7cf3-4bd2-ac8b-0fef46606c51 but yours will be different. Later on you will save this in a HCA global variable called HE_AccessToken.

You may want to start notepad and then cut these items and paste into a new notepad file for use later.



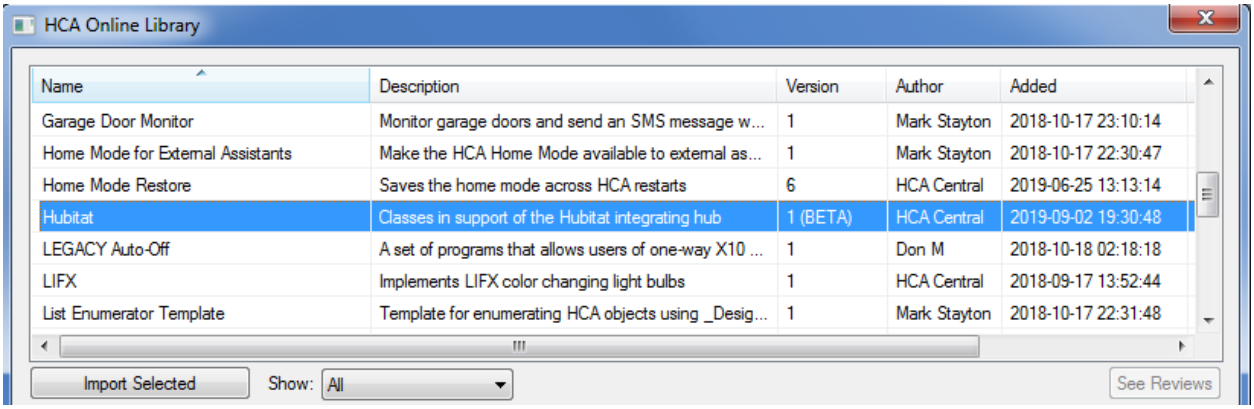
Scroll down to the bottom and press the “Done” button.



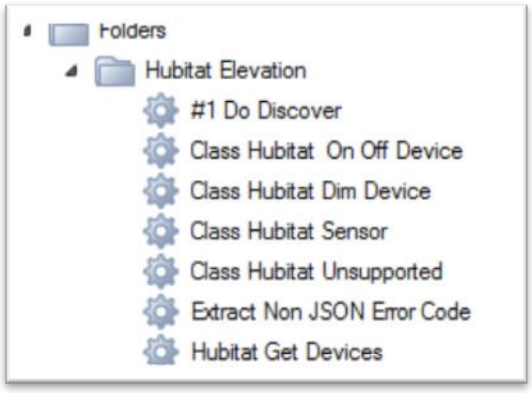
The display updates to show that the Maker API has been added.

Adding the Hubitat Package

For HCA to discover and operate all the devices controlled by Hubitat, you next add to your design a package from the online library. Open the library browser ("Design" ribbon category, "Library Browser/Import" button) and locate the Hubitat package and add that to your design.



The import creates a “Hubitat” folder containing these programs:



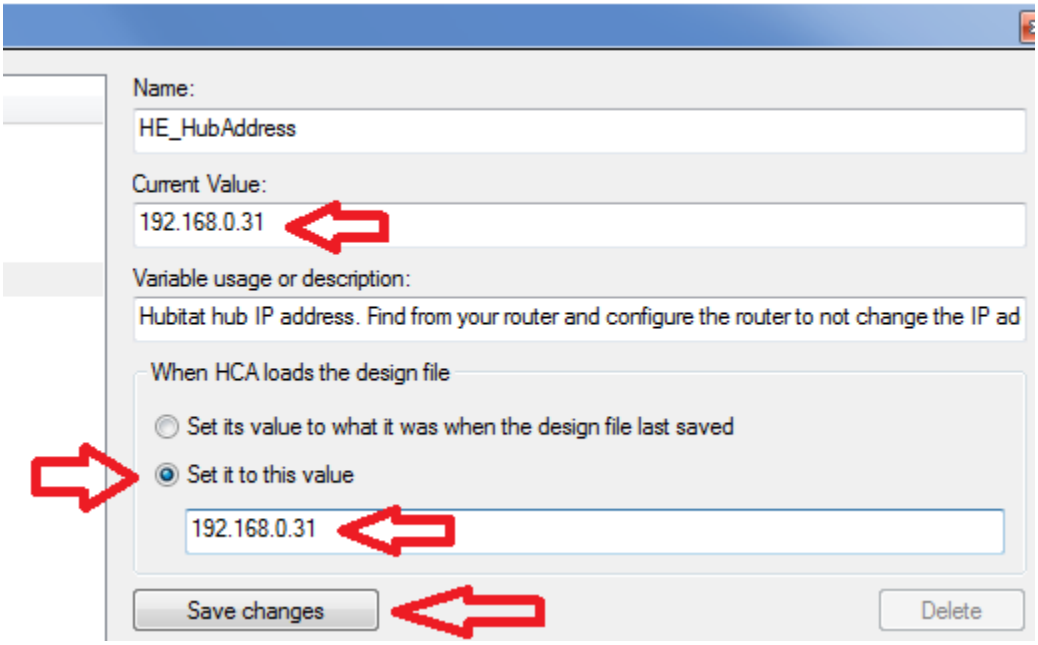
NOTE: there may be more or fewer of these programs as the package evolves.

Before doing anything else, open the Global variables inventory - “Control” ribbon category, “Tools” panel, “Variables” button.

There are three variables to update:

- HE_AccessToken
- HE_APICode
- HE_HubAddress

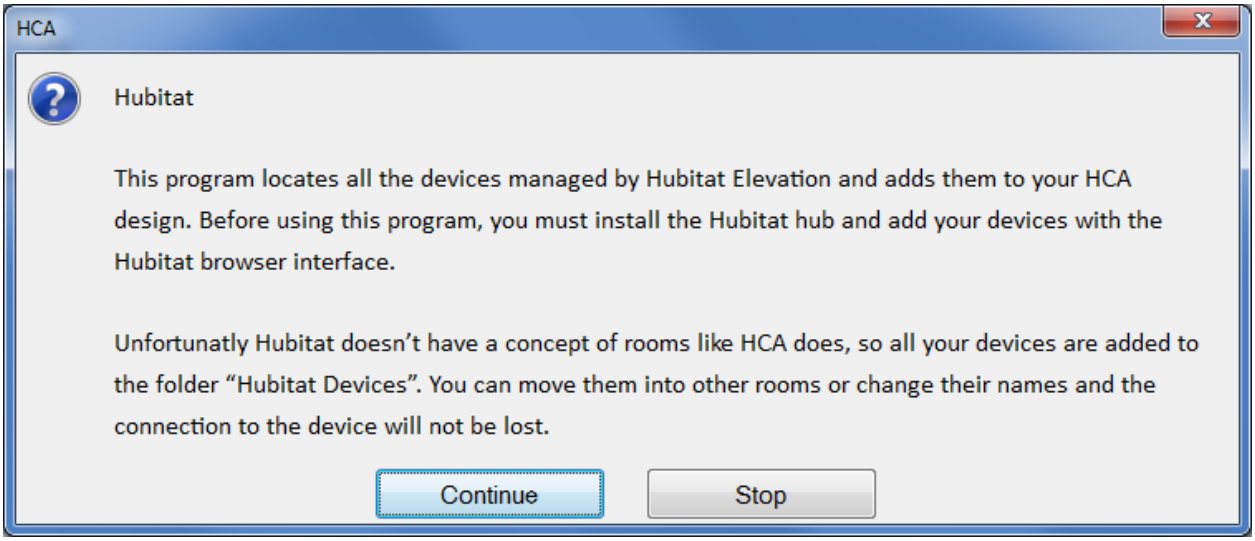
When adding the Hubitat package these variables are created, but all have empty values. Start with HE_HubAddress, and in the right section of the dialog change the values like this:



Enter the IP address in the “Current Value”, choose the “Set it to this value” option and then enter the same address in the edit below that option. Then press the “Save Changes” button.

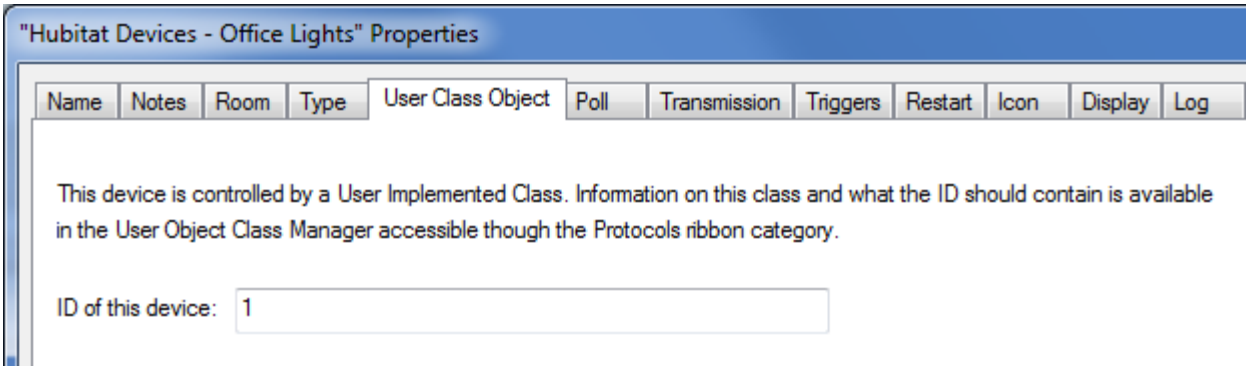
Set the “HE_AccessToken” and “HE_APICode” in the same way. Both the values came from what you cut from the example URLs when enabling the “Maker API” as described above. Make sure that you enter the values correctly. When all done, close the variables inventory.

Next, start the program “Hubitat Elevation - #1 Do Discover” to locate your Hubitat managed devices and to add them to your HCA design.



Press *Continue* and in a few seconds the devices are found.

The Hubitat has no concept of rooms, so all the newly discovered devices are added to a room called "Hubitat Devices". **You can move them to new rooms or change their names and the connection to the Hubitat device will <u>not</u> be lost.** The connection between the actual device and HCA is shown on the "User Class Object" tab of the device's properties.



That ID is the way that HCA identifies each device when talking to the Hubitat hub. You can't change the ID.

How to control the initial icon choice

Every device that is imported is assigned an icon. That icon choice is controlled by the kind of device it is. For example, a lock gets a lock icon, a motion sensor gets a motion sensor icon. For other types a similar choice is made. If you want to override these choices for your import, on the "Tags" tab of the "Hubitat Get Devices" program you can make that mapping from type to icon. The tag name is the device type, and the tag value is the icon name. If you don't use the tags to make changes, here are the icon selected for each type.

lock	Lock
presence	Room occupied
motion	Motion Sensor
contact	Door
water	Water Heater
switch / dim	Track Lights
nonDim	Relay A
button	CGrey
unknown	CRed

Using Hubitat devices

Once the devices are added you can control them like any other HCA device – ON, OFF, set to a percentage if dimmable, schedule it, use in programs, etc. Each action is handled by one of the class programs in the package.

For devices of the "Hubitat Sensor" class you can get their status.

Any devices assigned to the "Hubitat Unsupported" class will be non-functional as this class is a placeholder. See the next section.

NOTE: If a device goes offline, unfortunately HCA has no way of knowing this. When controlled or when requesting state, the Hubitat hub returns no error.

HCA Tech Note 613

Hubitat Device Types

The problem (and feature) of Hubitat is that it supports so much. Each device is represented by info that the server provides showing the “components” and “capabilities” of a device. Hubitat doesn’t provide a simple “type” – from some pre-determined list of available types – and isn’t immediately obvious what you can do with a device unless the capability list is examined. The HCA program that discovers your Hubitat devices attempts to understand the capabilities of each device and assigns that device to a class that handles devices with those capabilities.

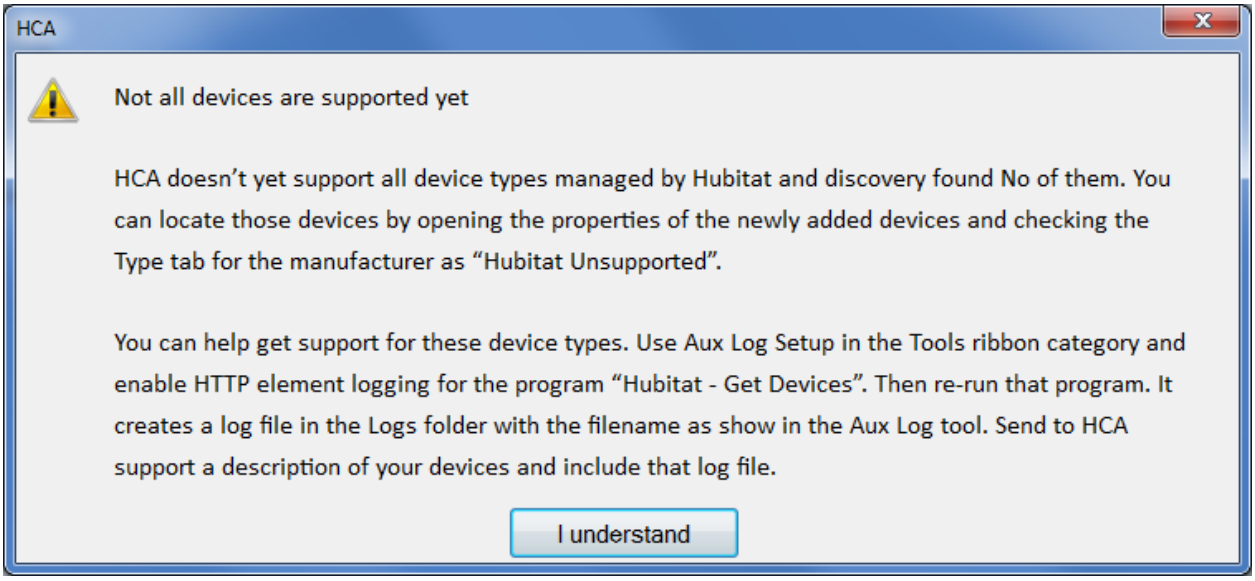
At this point these classes were created, and each Hubitat device is assigned to one of those classes. The classes are:

- Hubitat Dim Device
- Hubitat On Off Device
- Hubitat Sensor
- Hubitat Lock Device
- Hubitat Unsupported

The first two are for dimmable and non-dimmable switches, modules, etc. The third class is for sensors that have some sort of state - contact, motion, button, or water sensor. The Lock class if for locks, obviously. The final class is for everything else and is a placeholder so it doesn’t allow any control or status reading.

If during the Discover program’s execution any of the devices are managed by class “Hubitat Unsupported”, a second popup suggests what you should do.

HCA Tech Note 613



In the Hubitat package in the library is a program called "Check Batteries". When battery powered devices report their full status to the Hubitat hub they also report other properties, one of which is the battery level. Some devices also report temperature. If the Hubitat programs see these kinds of settings it saves them as tags on the device. For example:

Tag Name	Current Value
HE_type	Generic Zigbee Motion Sensor
HE_devType	Sensor
HE_devSubType	motion
HE_Battery	12
HE_Temperature	61.45

These are shown on the "Tags" tab of a device properties. In this case the motion sensor is reporting a temperature of 61 (better put on a sweater!) and a battery level of 12 percent (better change it soon!). You can schedule the "Check Batteries" program to run once a week or how often you want to, and it sends you email for any device with a battery level getting low. Of course, for this to happen you will have to configure HCA to be able to send email. Feel free to modify the program to change how it reports, or the battery level you want to report about.

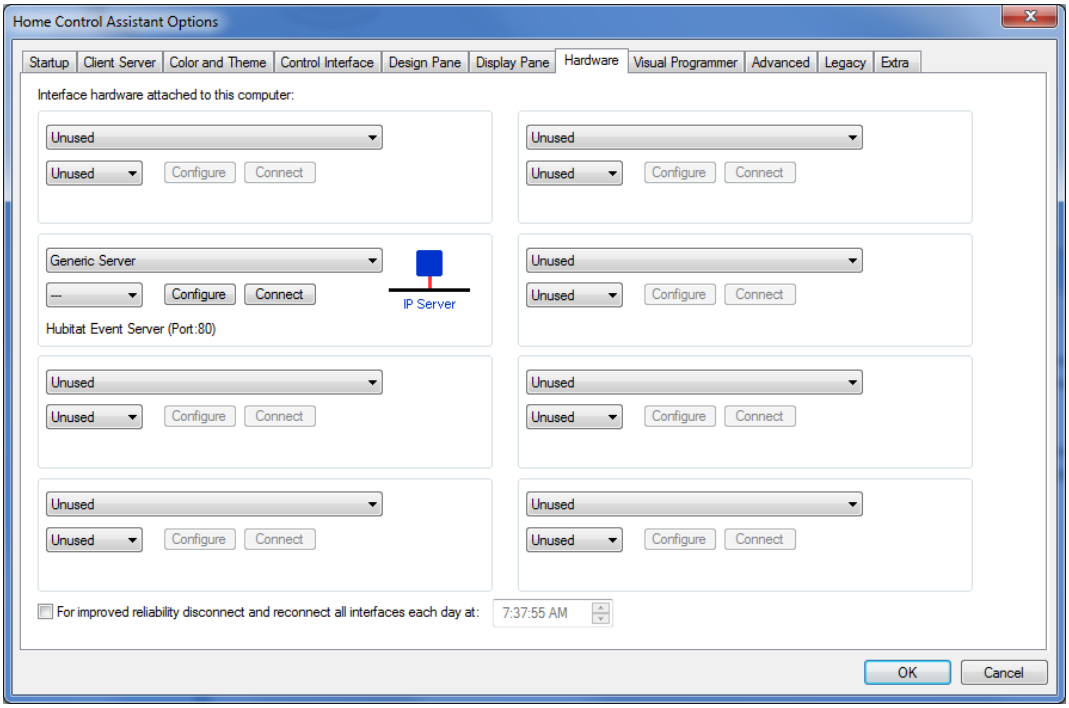
Because of these tags you can write your own programs to make decisions based upon their values – temperature for example – using the Tag functions in the Compute and Compute-Test programmer element expressions.

HCA Tech Note 613

Inbound Notifications – applies only to HCA 16.0.30 or later!

If you are running the latest version of the Hubitat "Maker API" then you can configure HCA to handle events that are generated by devices that the Hubitat Hub manages. Events like motion sensors becoming active, doors opening or closing, or switches being controlled on or off.

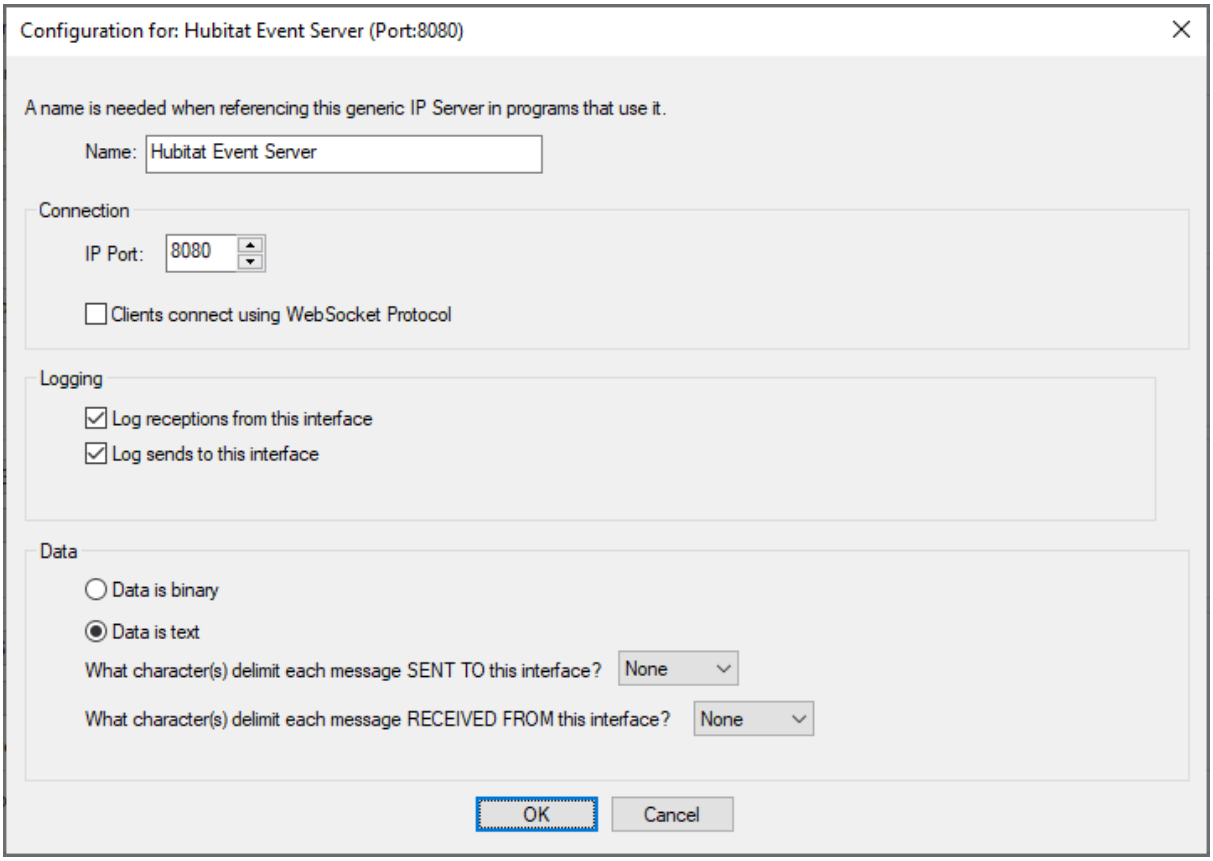
The first step is to configure an HCA Interface as a "Generic Server". Can be any of the eight interfaces, it doesn't matter which.



Press the Configure button to adjust the settings. The settings to use are: Name is "Hubitat Event Server", Port 8080, Data is text, receive delimiter is None.

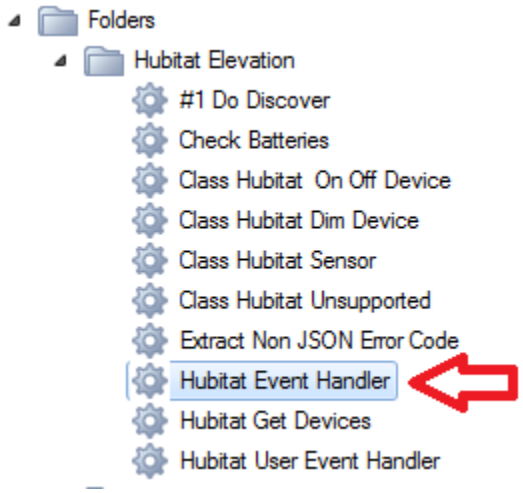
(Note: you can use a different port number other than 8080 if you want. In the section below where the Maker API settings are discussed you can see how you can choose a different port number.

Be careful here: There are other Generic Servers you may already be using to support other Hubitat access – see technical note #703. Don't make a mistake! This generic server must be called "Hubitat Event Server" and configured as described. This means you may have several Generic Servers in use simultaneously.

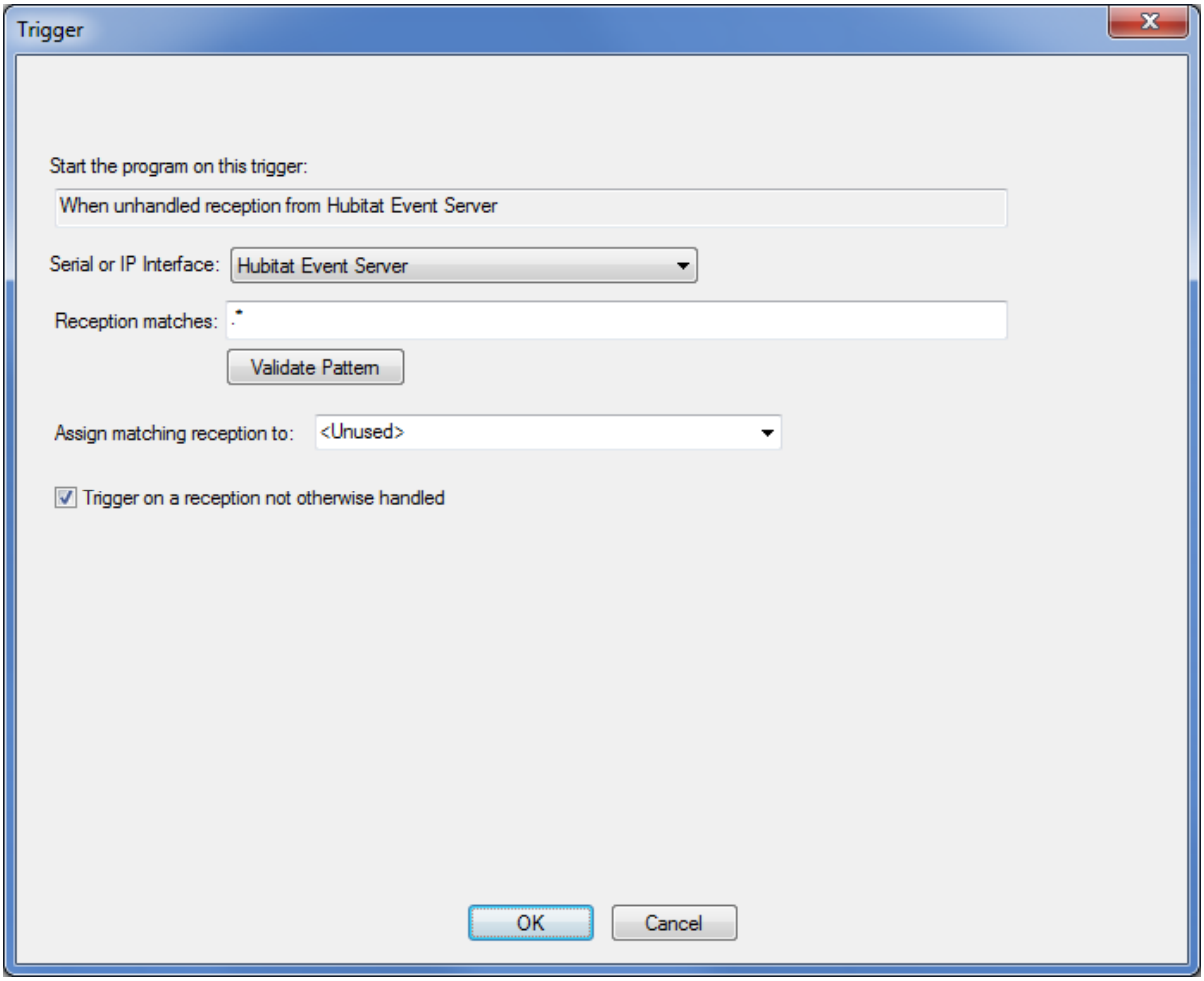


You can tick the *logging* checkboxes or not. That's up to you.

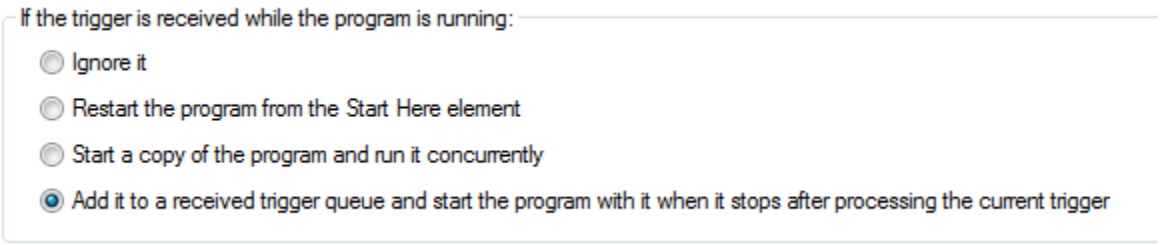
After importing the Hubitat package, you must next add a trigger on to the "Hubitat Event Handler" program.



Choose “Port Reception” for the Trigger type then select the interface as “Hubitat Event Server” which is what you named the interface in the interface configuration. Tick the box that says: “Trigger on a reception not otherwise handled”. All other settings can remain unchanged



In the main *triggers* tab, at the bottom, configure as:

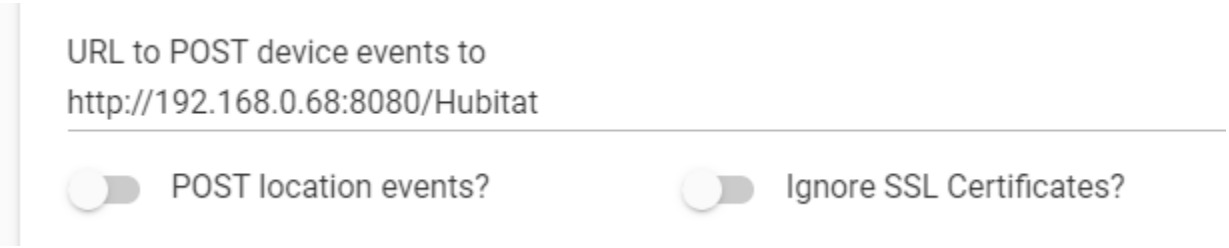


The next change is in the Maker API settings. As described at the start of this note, use a browser and connect to the Hubitat Hub, choose “Apps” from the left-hand menu, then “Maker API” and scroll down until you see this section:

The image above shows an IP address, yours will be blank. Enter the IP address of your **HCA Server** in this format:

http://<your HCA Server computer IP address>:<port #>/Hubitat

The port number is the port number you configured the "Hubitat Event Server" Generic Server interface in HCA as described above. Note that the port number is at the end of the 4-part IP address and is preceded by a ":".



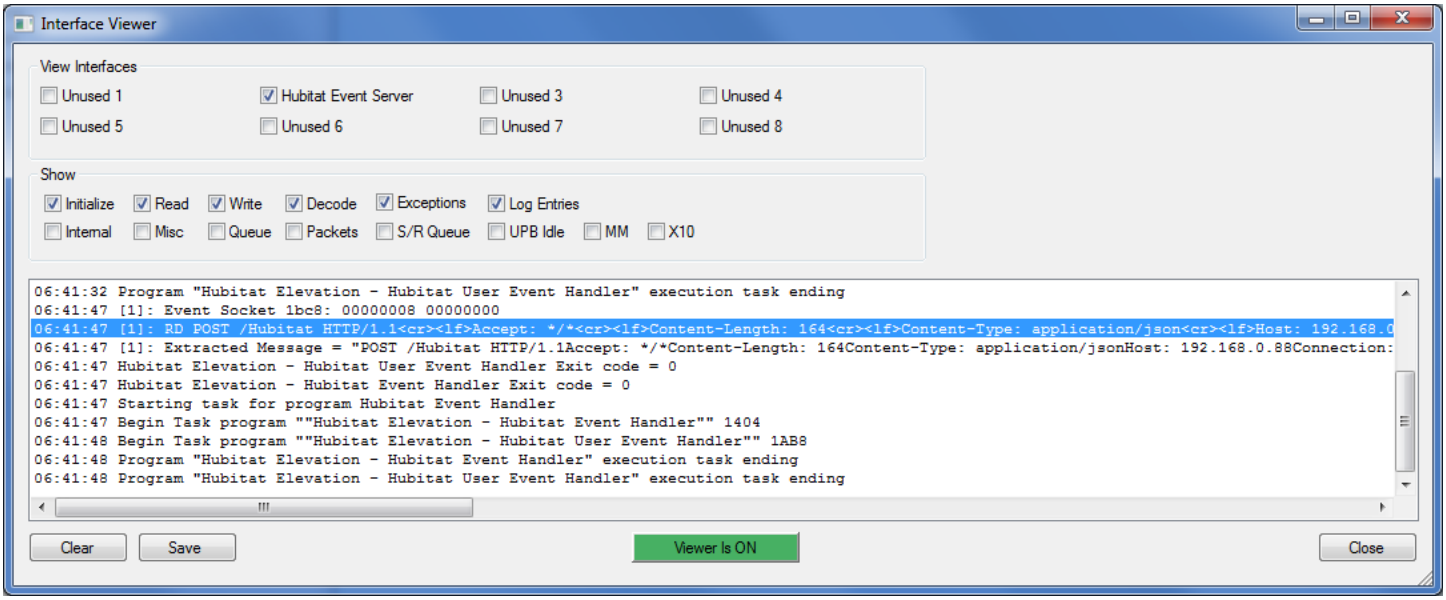
Just remember that here in the Maker API configuration you must use the same port number as in the HCA Interface setup.

After entering the "URL to send device events", then scroll down to the bottom of the Maker API configuration and press the *Done* button. This is VERY important!

To check if HCA is now receiving events use the HCA Interface Viewer. On the "Interfaces" ribbon category is an *Interface Viewer* button. Press that and a viewer window opens.

When something happens to a device that Hubitat controls you should see something recorded in the viewer if that interface is enabled in the viewer. In this example, Hubitat is the 2nd interface (in programmer world that is shown as [1])

If nothing shows in the viewer, make sure you have all the ports and IP addresses correct and that you have entered in the URL correctly into the maker API. Also check that the Hubitat hub and the HCA Server computer are on the same network.

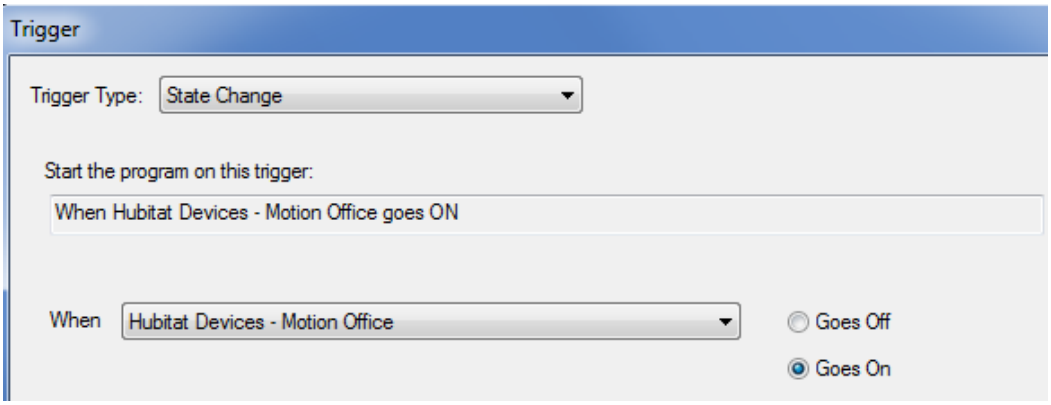

Interface Viewer
View Interfaces
Unused 1
Hubitat Event Server
Unused 3
Unused 4
Unused 5
Unused 6
Unused 7
Unused 8
Show
Initialize
Read
Write
Decode
Exceptions
Log Entries
Internal
Misc
Queue
Packets
S/R Queue
UPB Idle
MM
X10
06:41:32 Program "Hubitat Elevation - Hubitat User Event Handler" execution task ending
06:41:47 [1]: Event Socket 1bc8: 00000008 00000000
06:41:47 [1]: RD POST /Hubitat HTTP/1.1<cr><lf>Accept: */*<cr><lf>Content-Length: 164<cr><lf>Content-Type: application/json<cr><lf>Host: 192.168.0
06:41:47 [1]: Extracted Message = "POST /Hubitat HTTP/1.1Accept: */*Content-Length: 164Content-Type: application/jsonHost: 192.168.0.88Connection:
06:41:47 Hubitat Elevation - Hubitat User Event Handler Exit code = 0
06:41:47 Hubitat Elevation - Hubitat Event Handler Exit code = 0
06:41:47 Starting task for program Hubitat Event Handler
06:41:47 Begin Task program ""Hubitat Elevation - Hubitat Event Handler"" 1404
06:41:48 Begin Task program ""Hubitat Elevation - Hubitat User Event Handler"" 1AB8
06:41:48 Program "Hubitat Elevation - Hubitat Event Handler" execution task ending
06:41:48 Program "Hubitat Elevation - Hubitat User Event Handler" execution task ending
Clear
Save
Viewer Is ON
Close

Processing events yourself

When an event arrives from the Hubitat hub, the program named “Hubitat Event Handler” starts (that’s the program you added the trigger to). It processes the event to update HCA state. But you may want to process the event yourself. There are two ways to do that. Here are two methods.

Method 1: Responding to a device that changes state based upon an event

When a motion sensor sees something move, it sends an event that is processed by the “Hubitat Event Handler” program. That looks at the event data and sees that it applies to a motion sensor, so it changes the state HCA has for that sensor to be “ON”. You can create a program that triggers on a change of state for that device from OFF (inactive) to ON (active). Create a trigger on your program like this:



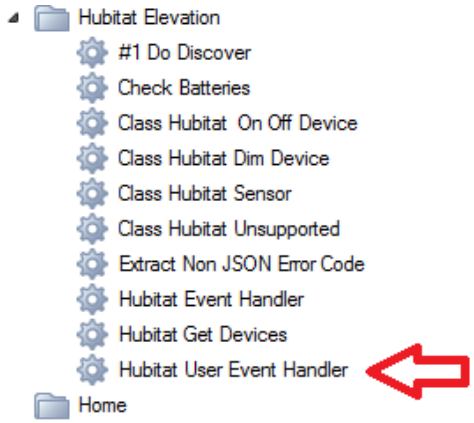
If you want to respond to the motion sensor going inactive, add a second “ON to OFF” trigger and then in the program, test for the trigger that stated the program.

In HCA everything is either ON or OFF. With sensors their state is ON/OFF but the meaning depends upon the sensor.

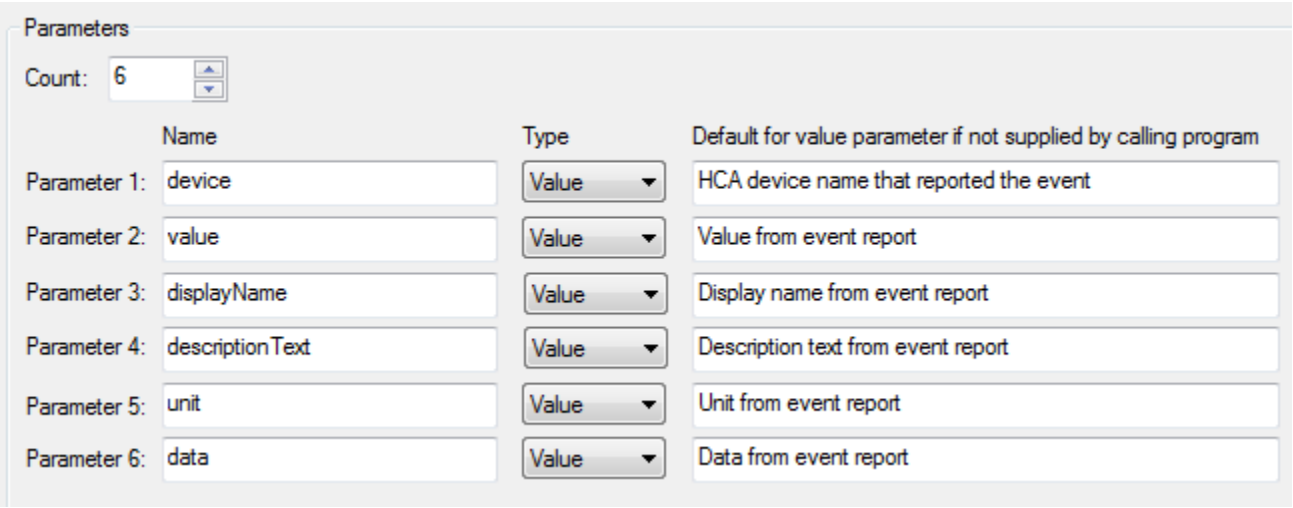
Device Type	OFF means	ON Means
Motion Sensor	Inactive (no motion)`	Active (Motion)
Contact Sensor	Closed	Open
Water Sensor	Dry	Wet
Presence Sensor	Not present	Present
Button	(other)	Pushed/double/held

Method 2: Responding to an event directly by a program

When the state change trigger works fine for events that change the state of the device, some events don't do that, and you must process the event yourself**. To do that find the program "Hubitat User Event Handler" in the package, copy it, and paste it into one of your folders.**



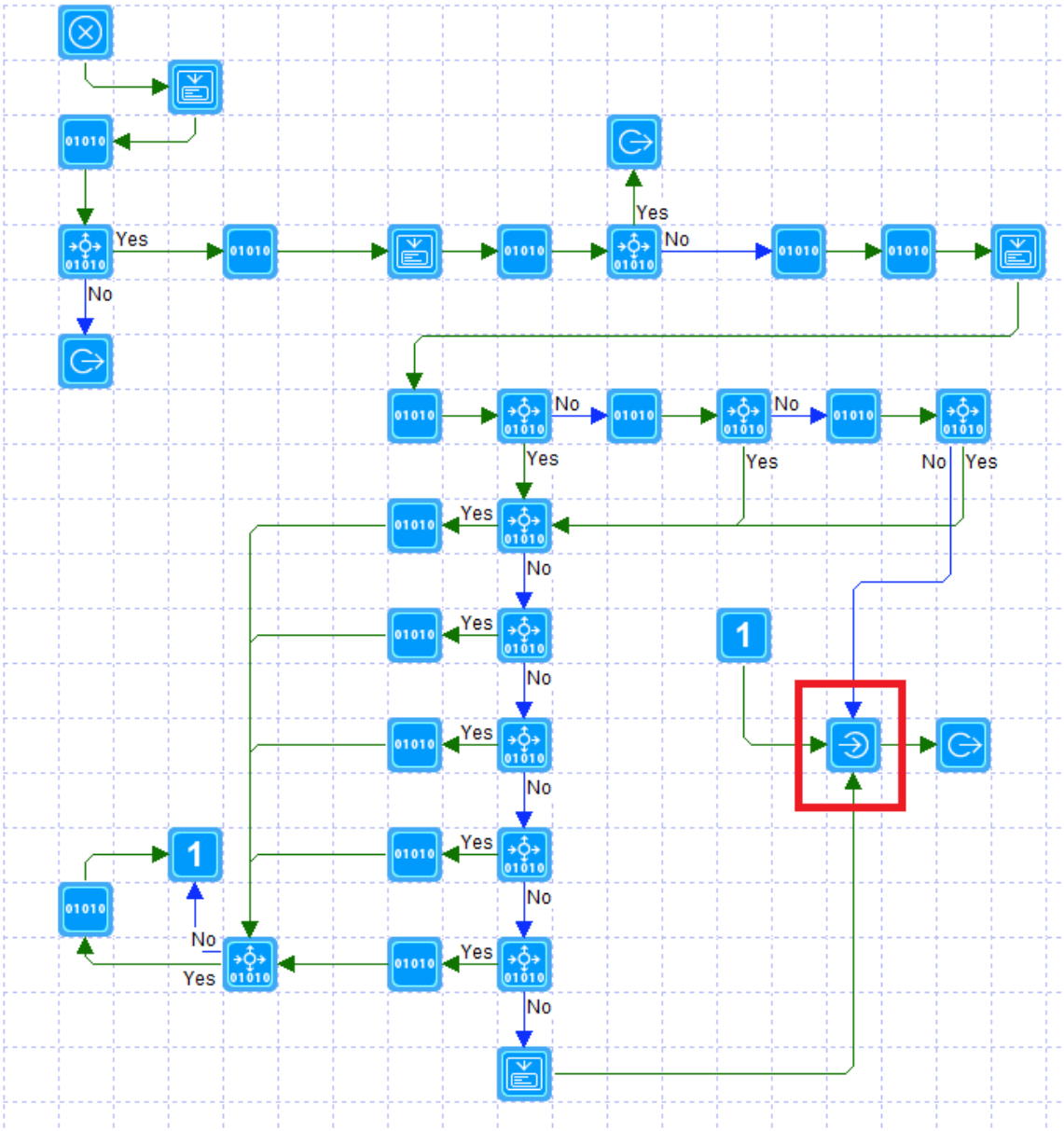
This is a parameterized program and is supplied with the elements of the event message.



The Event processing program in the package ("Hubitat Event Handler") has already extracted out all the elements of the event JSON, determined which HCA device it applies to, and then can start one of your programs so you also can process the event if you need to once the package handler is done with it.

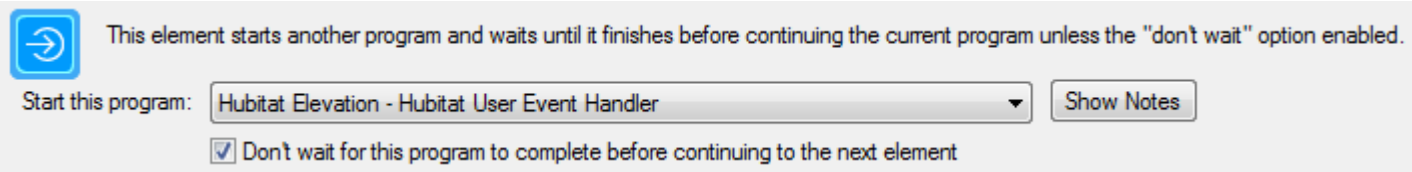
There is a placeholder program in the package called “Hubitat User Event Handler”. Don’t modify that program but copy it and paste it into one of your own folders with any name you want.

Next, open the properties of the package event handler program “Hubitat Elevation – Hubitat Event Handler” and find this element:

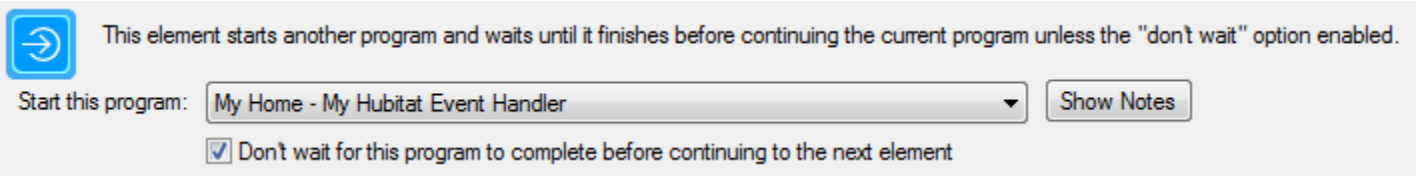


Change the properties of this Start-Program element to use your program and not the placeholder program in the package:

Change this:



To this:



In this example, the “Hubitat User Event Handler” was copied and pasted into the folder named “My Home” and renamed to “My Hubitat Event Handler”.

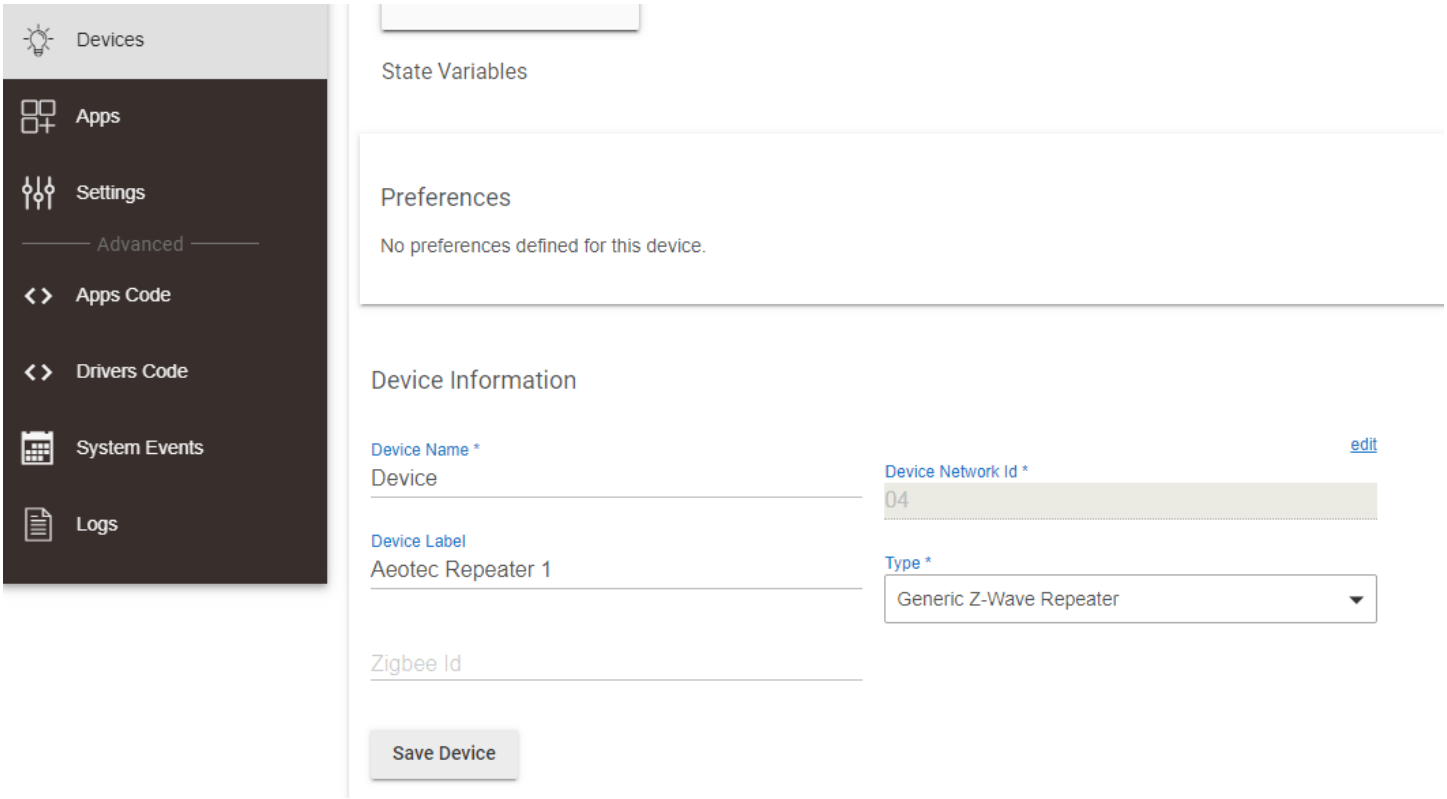
What your program does is up to you. You can test the various parameter values using the Compute and Compute-Test elements.

HCA Tech Note 613

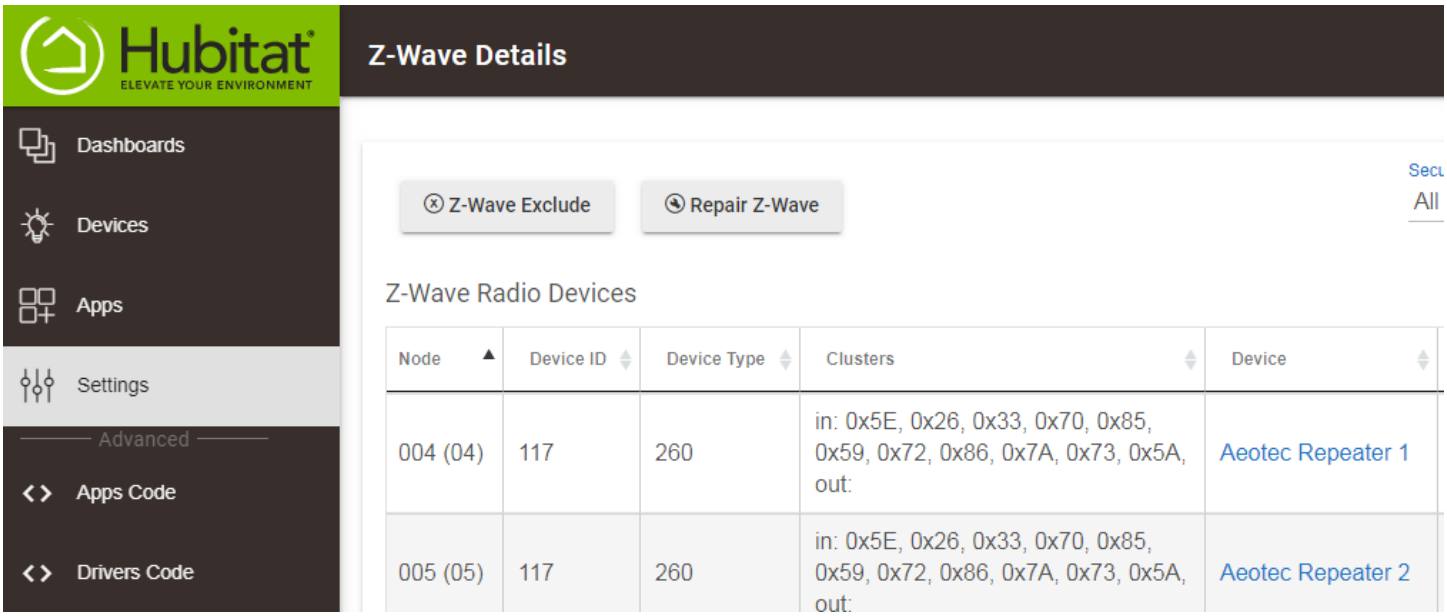
Using Hubitat Elevation with Z-Wave devices

The v5 Hubitat Hub has less than stellar Z-Wave sensitivity so it is a good idea to add at least 2 range extenders to your Z-Wave mesh before trying to add any devices. We have had good experience with the Aeotec Range Extender 6 repeaters and they are also cited often on the Hubitat Community as the best overall. Generally speaking, powered Z-Wave devices with repeater capabilities are insufficient to add reliability to your Z-Wave mesh when they are used with the HE hub, so adding these extenders is a good idea especially if you want to control Z-Wave locks. You will need to place one range extender within two feet of your hub and the other(s) within 10-15 feet of the devices you want to control.

When you add these devices to your Z-Wave, they will be discovered as generic devices. Be sure to change the Type from "Device" to "Generic Z-Wave Repeater". Make the Device label anything you want.



Next, repair your Z-Wave network from the settings/Z-Wave Details page on your HE hub:



After the repair finishes, it is best to wait 24 hours before trying to add devices to allow the mesh to fully re-establish itself.

It is also a good idea to put fresh batteries in the device you want to add if it is battery powered. We have had situations occur with lock devices where their full capabilities were not recognized if the batteries weren't fresh. In one case it led to the lock being successfully added but subsequent commands not being recognized, and in another case, the lock was added but its auto-refresh capabilities weren't discovered. Put in fresh batteries to be sure.

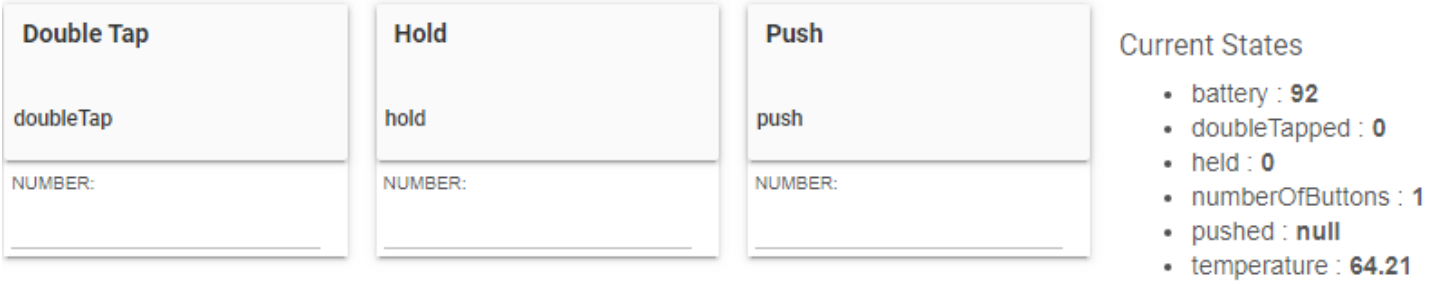
HCA Tech Note 613

Using Buttons with Hubitat Elevation

Zigbee buttons such as the inexpensive Samsung SmartThings Button are great little devices to have in Hubitat because not only can they be used to initiate programs in HCA with various tap and hold options, but they can also report their temperature. HCA recognizes these buttons as sensors and creates a special icon for them that shows the last press event as well as the button's current temperature:

However, in order to make these buttons useful, some programming needs to be done using the Hubitat Rule machine. If you don't already have this installed, it can be found in the built-in apps. From the Hubitat browser interface, select "Apps" in the left menu, then press the "Add Built-in App" button and find "Rule Machine" and select it to add.

The problem is this: buttons have three "state variables" associated with the push actions they recognize: "pressed","doubletapped", and "held".



The problem is that when a button is manipulated, the associated State gets set to a 1 and it stays that way forever. This makes it impossible for HCA to request a status that returns the last press action or for you to know that the button got

tapped in any manner. So how to solve this? One way is to manually reset the state by performing the action from the device tab. Entering a “0” in the NUMBER field above for the desired state to reset, then performing the action is one way to do this but obviously not ideal.

Enter the Rules engine which is really the primary programming feature of Hubitat. Using the engine we can trigger on a press, hold or doubletap event, then create actions that reset the state via custom command after a 3 second delay.

Creating these rules is very tedious but not hard. Here are the steps.

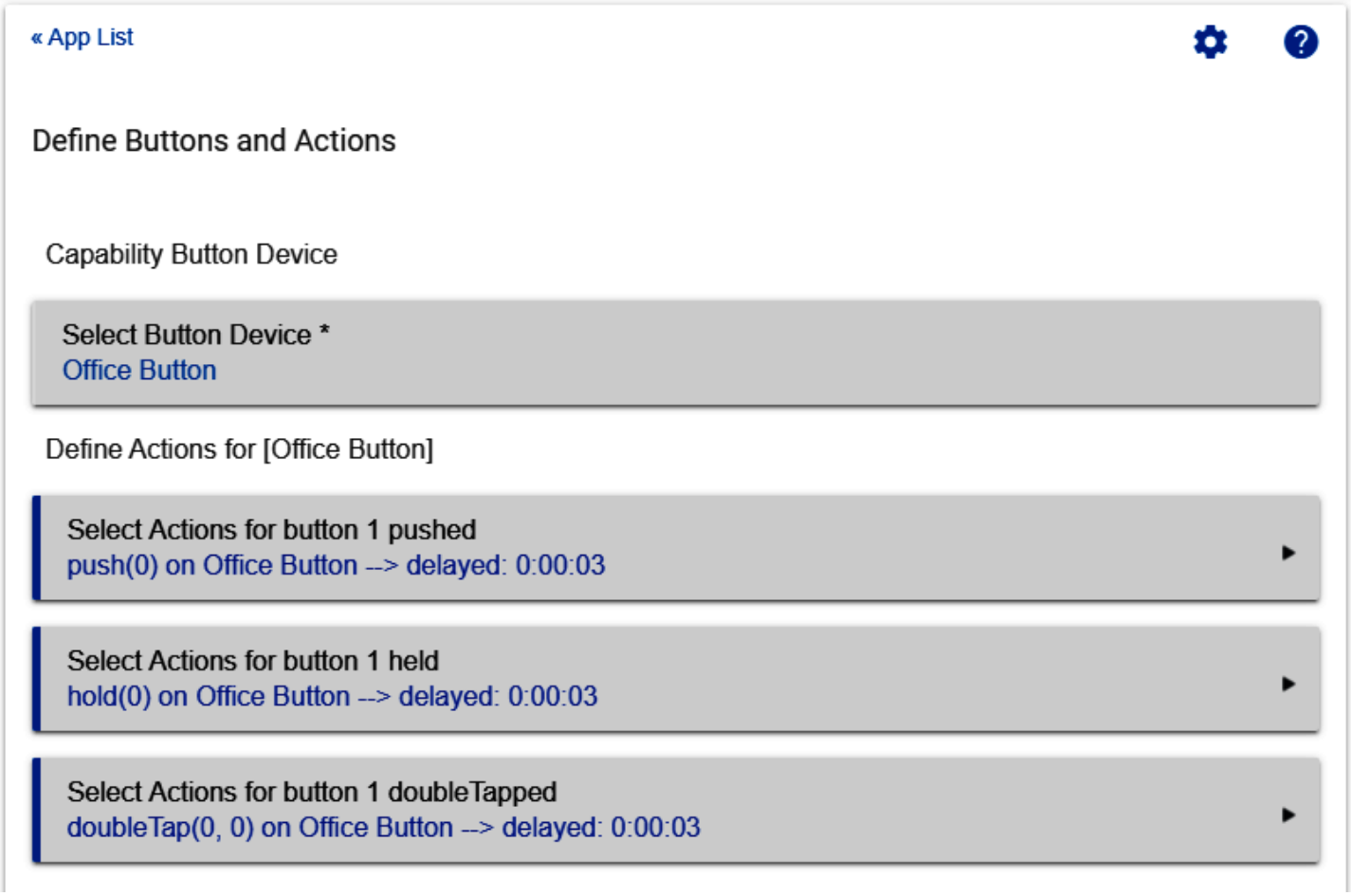
- In the “Rule machine” app click”Create new rule”
- “Name this rule” (click to set). [Click] Enter the name "Reset button state" [Enter]
- “Select Trigger Events” (click to set). [Click]
- “Select capability for new Event Trigger” (click to set). [Click]. Choose “Button Device” (not “Button”)
- “Select Button Device” (click to set). [Click]. Tick box next to your button [Update]
- “Prefill all button actions for” (click to set). [Click]. Tick all three. Click elsewhere and the display updates.

- “Select Actions for button 1 pushed” (click to set) [Click]
- "Select Action Type to add" dropdown, select " Set mode or variables. Run custom action"
- "Select Which Action" dropdown. Select "Run custom action".
- "Select capability of action device" (click to set) [click] Choose "button"
- "Select Button devices to act on" (click to set) [click]
- Tick box next to your button [Update]
- "Select custom command" dropdown. Choose "push"
- "parameter type" dropdown. Choose number
- "integer value" Enter 0 [Enter]
- Tick "delay"
- Enter 3 under seconds. [Enter]
- "Done with this action" [click]

- [Done with Actions for button 1 push" [Click]

Repeat this section starting with "Select actions for button 1 held" and then repeat again for "Select actions for button 1 doubleTapped" doing the same thing but in the "Select custom command" step, choose not "push" but the appropriate command.

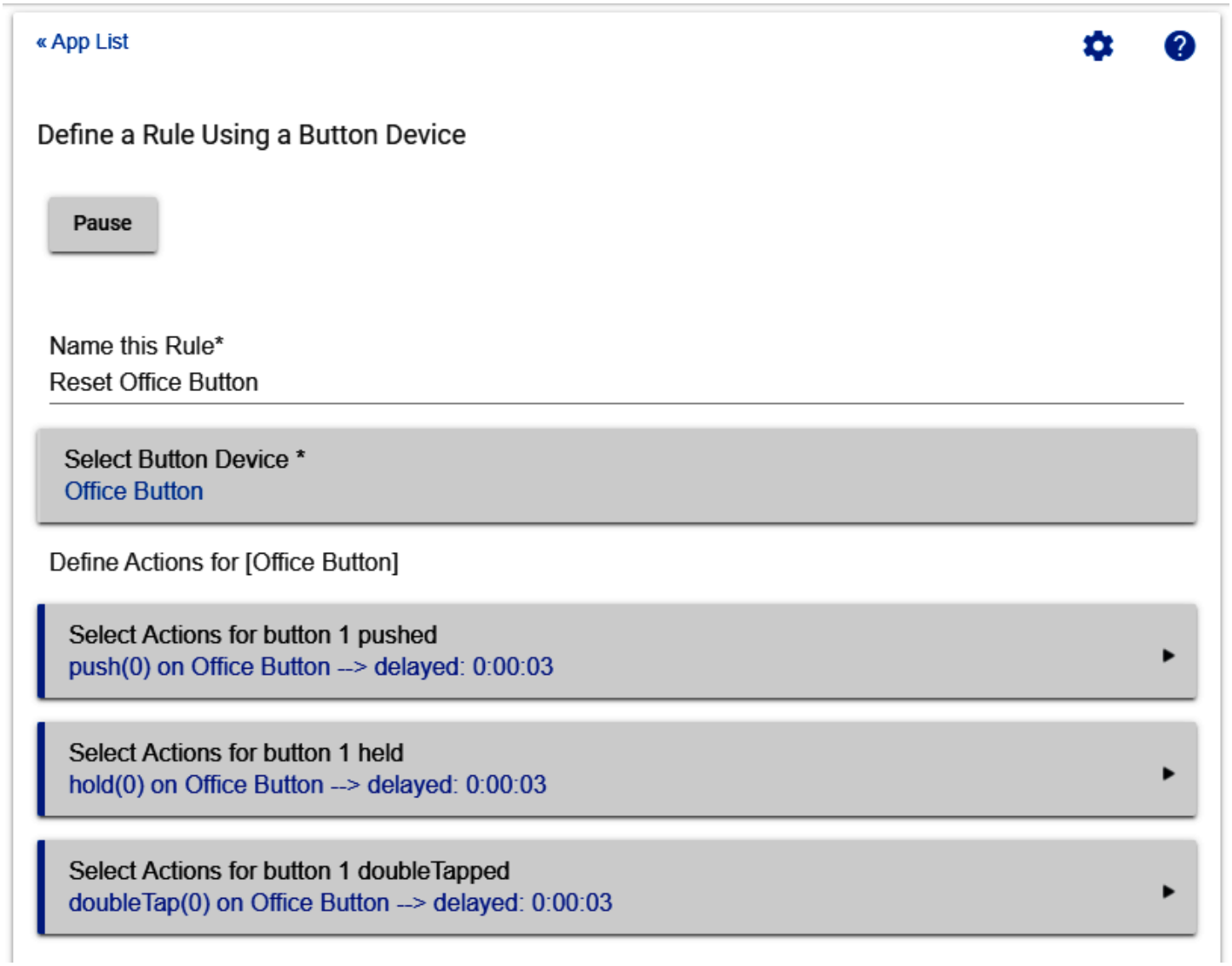
When all three are dpone, this is what you should see:



"Done" [click]

"Done" [Click] This should get you back to the Apps list and you should see your new rule listed below the Rule Machine.

If you click on that riule you should see this:



OK, that was tedious and I bet you appreciate the HCA visual programmer even more.

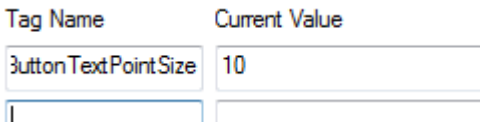
When the button is pressed, double pressed, or held, HCA is notified and then 3 seconds later the button state clears and HCA is notified again.

So how does the automation map into HCA? Here's how:

- The button is pressed, double tapped or held
- HCA gets an event from Hubitat and the built-in event handler stores the action in the button's "HE_LastAction" tag, It then changes the button's state to "On", changes the button's icon to its "On" depiction, and updates the text displayed on the button.
- KEY POINT: The "On" state change causes any programs with a state change trigger for that button "off" to "on". They program can query the HE_LastAction tag to find out what event occurred and act accordingly.
- Three seconds later, the Hubitat rule kicks in and resets the state.
- HCA gets another event from this, changes the button's state to "Off" and changes the button's icon to its "Off" depiction.
- The "Off" state change causes any programs with a state change trigger for that button "on" to "off". They program can query the HE_LastAction tag to find out what event occurred and act accordingly.

TIP: In the package there is an example program "Hubitat respond to push". Don't modify the program but copy it and paste it into one of your own folders. Then add state change triggers for your button. You can modify the program to do whatever you want when the button is pressed, double tapped, or held.

TIP: If you want to change the size of the text on the button, add a tag to the "Hubitat Event Handler" and "Class Hubitat Sensor" programs. The tag name is "ButtonTextPointSize" and the tag value is the point size. For example, 10 point text.



Enjoy using these versatile Hubitat devices!

##end##