**Tutorial : IFTTT and VERA Integration with IFTTT**

**Introduction**

* IFTTT ( <https://ifttt.com> ) is a well-known internet software service which enables the creation of basic “applets” tighting together sources of events and actions on various objects or other services. So IFTTT applets react to “triggers” and perform “actions”. There are tons of supported triggers and actions like sending notifications , sending emails, editing documents on your preferred cloud storage, acting on IOT devices etc.
* VERA so far is not one of of the supported IFTTT “services”. GetVera company is working on an VERA integration as an IFTTT action part ( so IFTTT can act on a device or run a scene on vera ) but there is no support for incoming data integrations where VERA would act as a “triggers” in a applet. The focus of this tutorial is to show how easy it is to use VERA as a trigger using ALTUI new IFTTT integration.
* ALTUI supports since a while the concept of “Data Storage” integration to enable a user to send some data externally, via a web service call, whenever a particular device variable changes. Supported integrations so far were thingspeak , emoncms, Datayours and the list is open to extension by third party. In the latest ALTUI release a new internal “Data Storage” integration is added for IFTTT thus enabling any chosen device/variable change to trigger some IFTTT actions

**Typical example of accomplishments using VERA as a trigger**

1. Track the changed values of a device over time in a google sheet to get some history in a free/low cost/zero server / zero DB fashion
2. Send an email, or a mobile device push notification whenever some device value changes.

This document is a step by step tutorial, with screen shots, that will walk you through the use case #1 here above.

**Prerequisites**

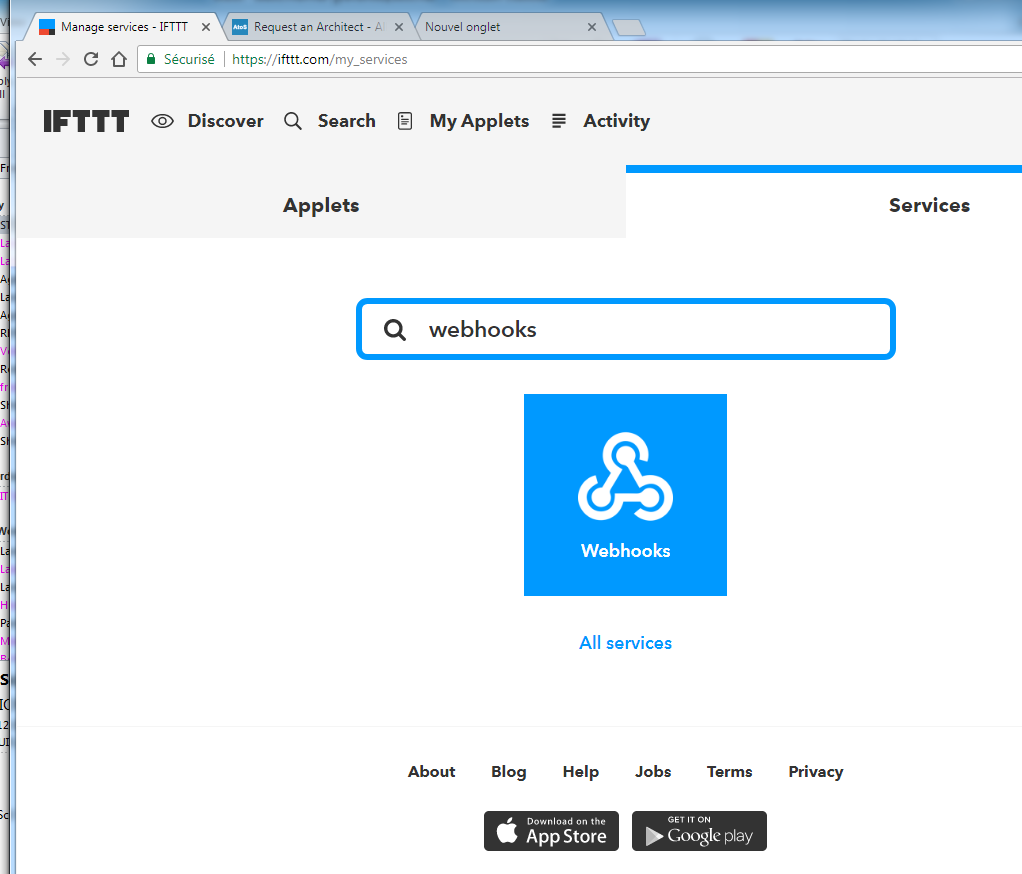
* Vera or openLuup
* Running ALTUI plugin <http://apps.mios.com/plugin.php?id=8246>
* An IFTTT account
* A Google account with a google drive folder where we will store a google sheet

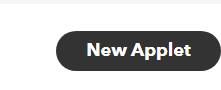
# 3 fundamental steps

1. Configuration of IFTTT : where we configure IFTTT and prepare the necessary IFTTT applet
2. Configuration of ALTUI : where we configure ALTUI to send data notifications to IFTTT
3. Fine tuning : some improvements

## Configuration of IFTTT

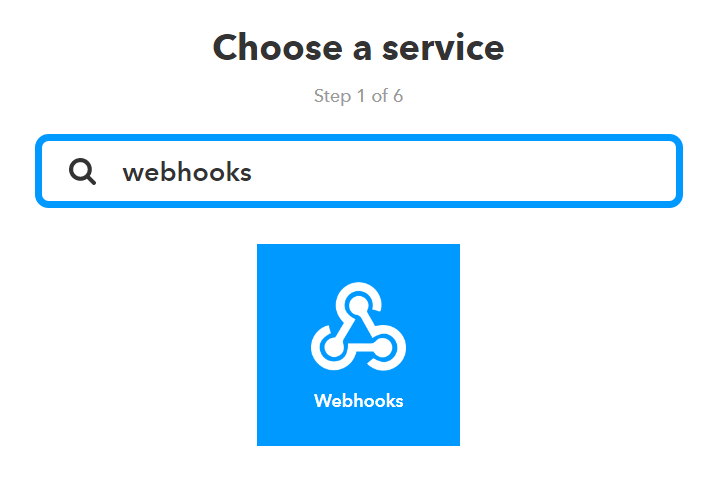
* First you need to have an IFTTT account <https://ifttt.com/discover>
* We will need to use the IFTTT service webhooks. For this you can go to My Applets and search for “WebHooks” and configure it



* Then go to “My Applets” and create a new applet clicking on the New Applet button
* You will be presented with a screen like this showing the basic structure of an applet with a **THIS** part and a **THAT** part.

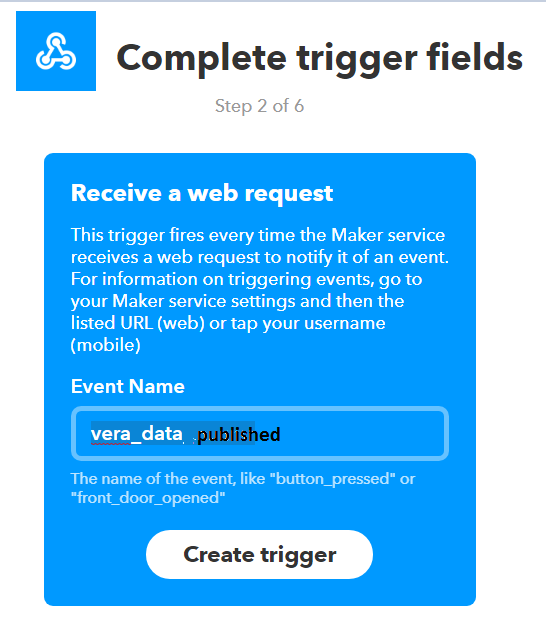


### Step1 – prepare the applet

* In our case , we want to receive a web service from VERA so the **THIS** part will be a WebHook. Click on THIS, then type WebHook in the search field , then click on WebHooks bleu square  
  

### Step 2 – configure the webhook

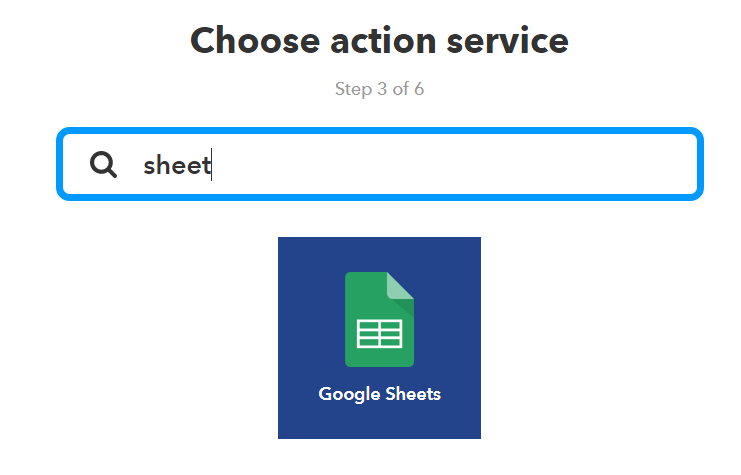
* Click again on the “Receive a web request” bleu square and IFTTT will ask you for an event name. you can choose whatever you want here. You can have different event name per devices, or use the same if you just want all devices variable changes to go through the same action. Event name in the same google sheet can enable groupings differently, you can even have different files per event name. This is up to you. In my example I arbitrarily chose “vera\_data\_published” and all will be in the same file.
* Remember that event name it will be useful later

  
then click on “Create trigger”

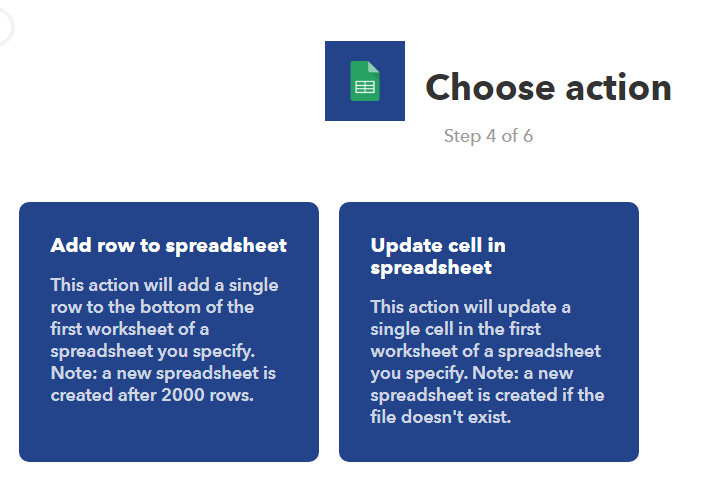
* Now the Applet THIS part is complete and we still miss the THAT part. So click on the “That” word  
  

### Step 3 – configure google sheet action

* Choose a service ( if you are part of VERA IFTTT beta program you may see the VERA action service but this **is not what we are interested here** )
* Search for google sheet and click on it

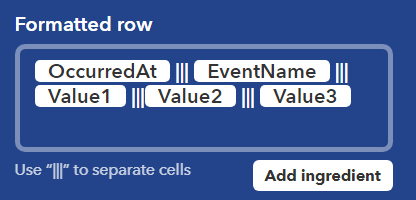


### Step4 - some simple formulas in the action

* We want to add a new row in a google sheet, every time we get a device variable change ( for the chosen devices in ALTUI, we will see that part later ).  
  
* Click on Add row to spreadsheet
  + By default the action is configured to use a google sheet named “IFTTT\_Maker\_Webhooks\_Events”, located in your google drive in a folder called IFTTT/MakerWebooks/{{EventName}} where {{EventName}} is the name you have chosen in step 2 so it will be IFTTT/MakerWebooks/vera\_data\_source
* **Go ahead and create a google sheet of that name in that folder in your google drive. IFTTT would create the file alone but we are going to do some things in it later so we need to create it now by hand :**

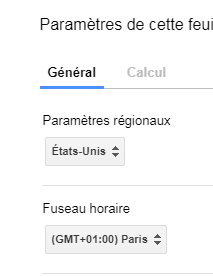
**File : /**IFTTT/MakerWebooks/vera\_data\_source/TTT\_Maker\_Webhooks\_Events

* + The interesting part is the row that will be inserted in the sheet. It will allways be a new / last row of the first sheet tab . The row cells values will be what you put in the Formatted row field



* + You can add as many cells as you want, providing you separate each cell definition by 3 vertical pipes : like “**|||**”
  + This is the default but we can improve it quite a bit
    - **OccuredAt** is a textual field representing a date/time like “December 4, 2017 at 09:33AM” but that format is not very good in sheets for graphs or filtering. Fortunately you can use any google sheet formula. So I basically use 2 cells and formulas to translate into a valid date and a valid time:
      * Date : DATEVALUE(REGEXREPLACE( "{{**OccurredAt**}}"," at .\*",""))
      * Time: TIMEVALUE(REGEXREPLACE( "{{**OccurredAt**}}",".\* at ",""))

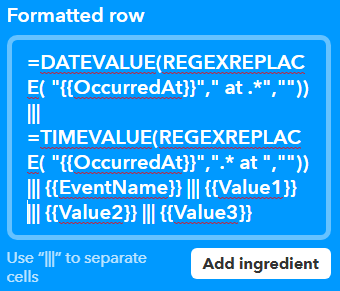
|  |
| --- |
| * NOTE: for non-English speakers, it is possible that this does not work because your google sheet locale is not set to English. We have no control over strings sent by IFTTT so to make it work, set your google sheet locale to English/USA. This can be done in the File / Parameters of the sheet menu command. |



* + - For the Value1, Value2, Value3 , they are determined by ALTUI integration code. ALTUI will send :
      * **Value1** = the altuiid of the device ( like 0-123 )
      * **Value2** = the service”:”variable that has changed like “urn:micasaverde-com:serviceId:DoorLock1:Status”
      * **Value3** = the new value taken by the variable

So the formula I put in Formatted Row field ends up being:

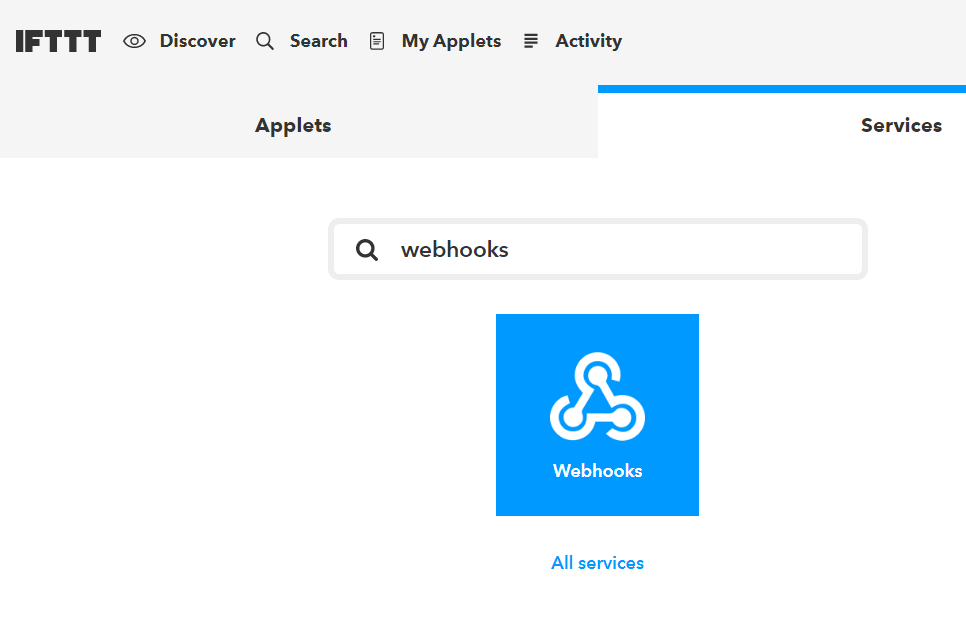
=DATEVALUE(REGEXREPLACE( "{{**OccurredAt**}}"," at .\*","")) ||| =TIMEVALUE(REGEXREPLACE( "{{**OccurredAt**}}",".\* at ","")) ||| {{**EventName**}} ||| {{**Value1**}} ||| {{**Value2**}} ||| {{**Value3**}}

* 

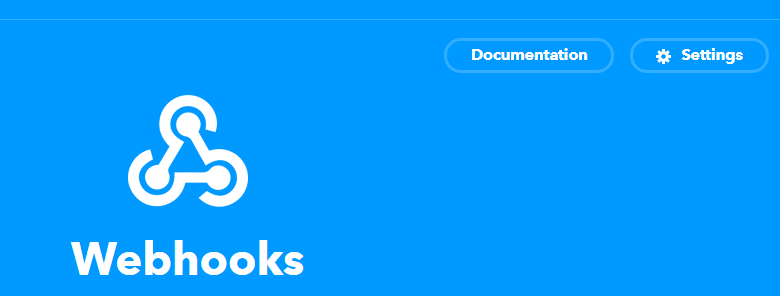
Then Save.

### Step5 – remembering the WebHook API key

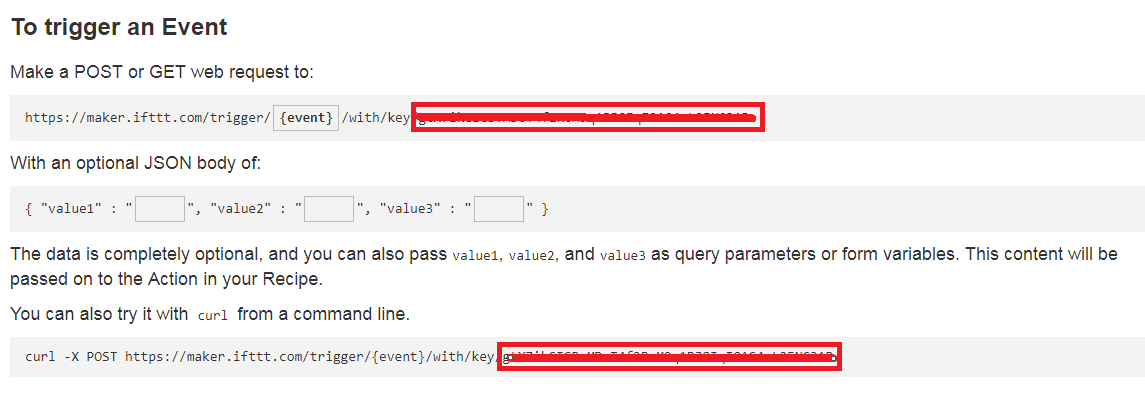
* Before going into ALTUI, we need to figure out what is the key that IFTTT allocated to use for the webhook. So Go again into the IFTTT Services screen and search for webhooks



Then click on it and click “**documentation**” on the following screen



Now you get into an important screen where you can see the webhook key that you need to use



Copy the key in the read area somewhere, you will need it later.

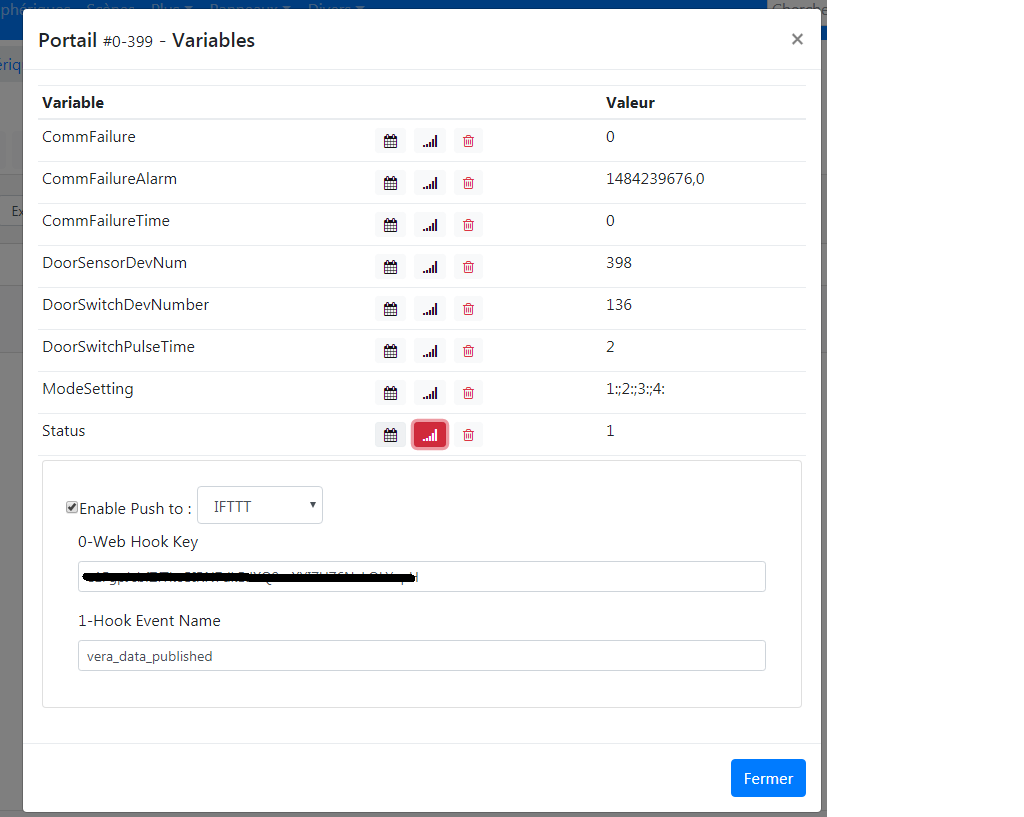
## Configuration of ALTUI

### Step1 – configuring the data push parameters

In ALTUI, the process is very simple. First you identify and go to the control panel of the device you want to use as the IFTTT trigger and click on the “histogram” icon to open the Data push properties.

* Here you select IFTTT
* You enter the Key that you have captured in the previous step
* You enter the event name that you have chosen when you configured your new webhook applet

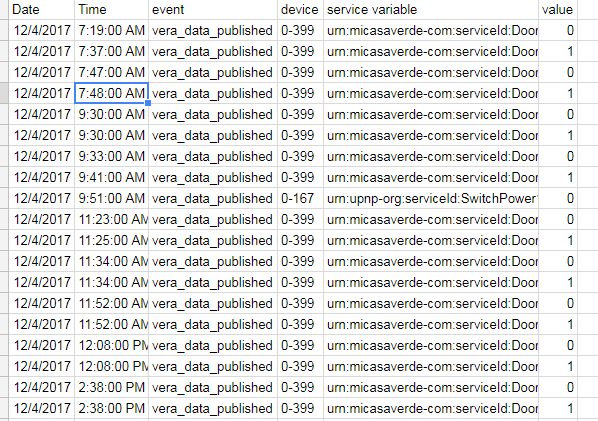
|  |
| --- |
| **NOTE**. Do not forget to CLOSE THE HISTOGRAM icon to save the changes |



Then click OK and **do a Luup Reload** to be sure all changes are taken into account.

From there on, any changes on this variable will trigger a web service to IFTTT which in turns will trigger your applet action, so for this example it will add a line in a google sheet

Here is the resulting sheet.

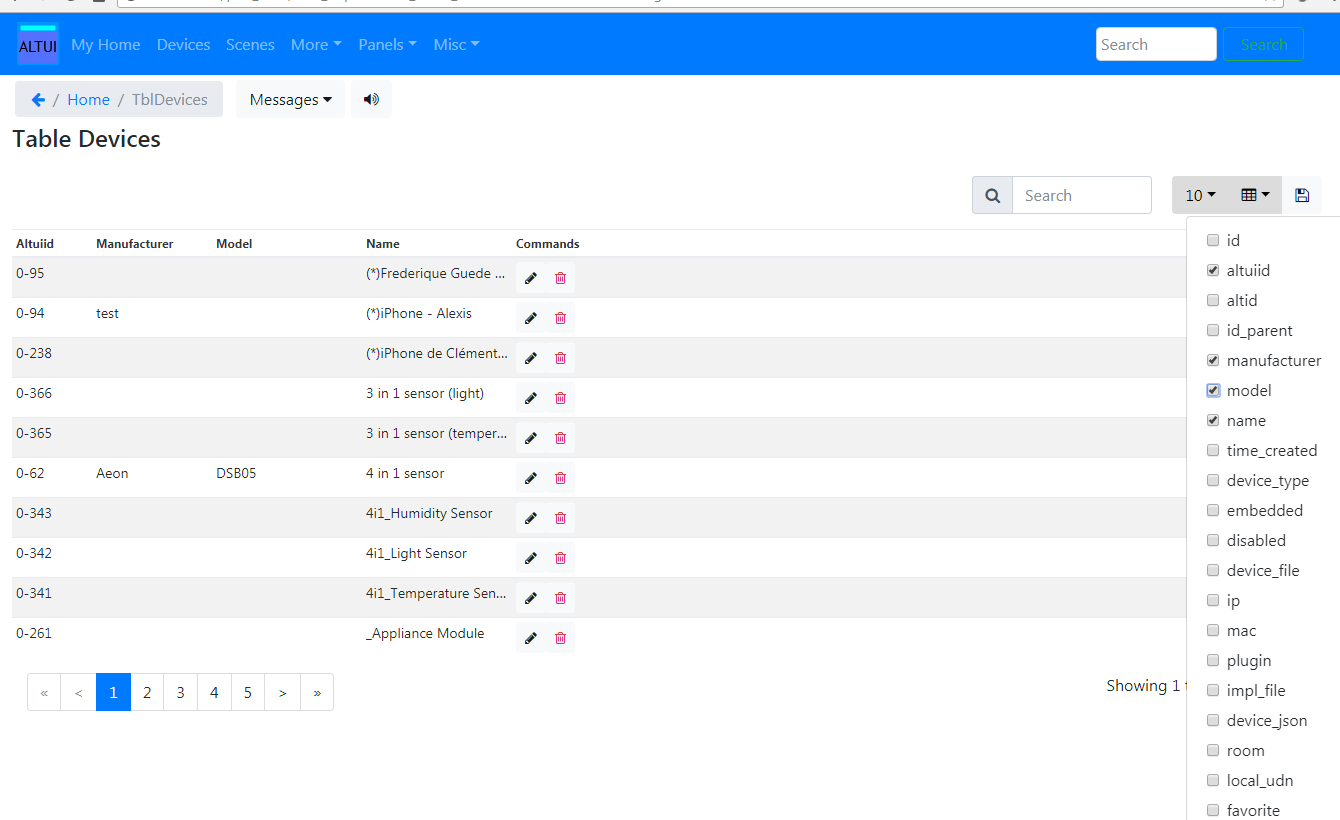


## Some additional Fine Tuning

Out of the sheet there is a nice enhancement needed so that we display the device name instead of just the device ID. We need a mapping table from device altuiid to device name.

### Step1 – gather a table : device altuiid ⬄ device name

* For this we can go into ATLUI in the page More / Table Devices. Select ALTUIID & manufacturer, model and name fields in the field selector to show these columns. The “Name” is therefore the 4th field in each row



Then select “All Rows” in the batch size selector and click on the copy to clipboard button on the right.

### Step2 - creating a mapping table in the sheet

* Then go into the google sheet file and add a second tab ( must be second, IFTTT will fill in the first table always ) , you can name it “Devices” for instance and select A1 cell then CTRL+V ( paste ). The following similar textual CSV data should be pasted in the cell.

"altuiid","manufacturer","model","name","Commands"

"0-94","test","","(\*)iPhone - Alexis",

"0-366","","","3 in 1 sensor (light)",

"0-365","","","3 in 1 sensor (temperature)",

"0-62","Aeon","DSB05","4 in 1 sensor",

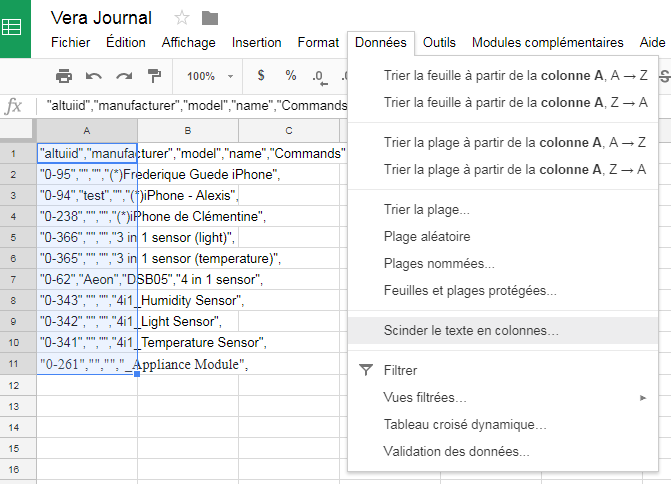
"0-343","","","4i1\_Humidity Sensor",

"0-342","","","4i1\_Light Sensor",

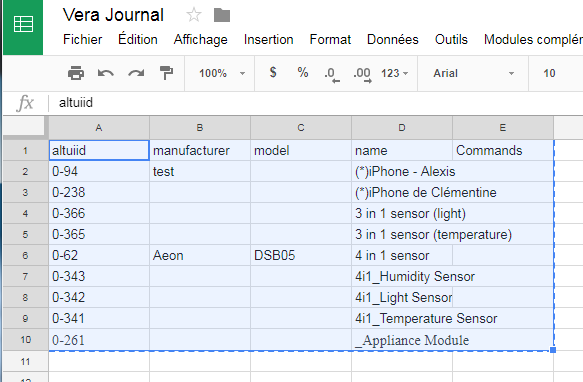
"0-341","","","4i1\_Temperature Sensor",

"0-261","","","\_Appliance Module",

Then select the whole range of cells and go to **Data / Split text to columns command**



That should give a nice table like this



* Which means the table can now be used in a VLOOKUP() function so that , from a altuiid, we can find the device name. assuming this whole table is located in a tab called “devices” and it spans from A2 to D161, then the following function can give us the device name

|  |
| --- |
| =VLOOKUP(TRIM(<the altuiid given by the webhook>), devices!A2:D161, 4, false) |

### Step3 – updating Google Sheet

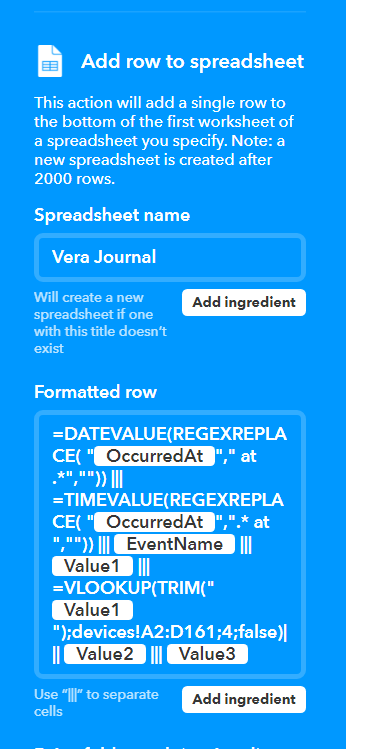
* Now we will have to make sure the IFTTT action is properly using formula, so we go back to IFTTT and edit our applet property this way. See the Formatted row value which includes a new field ( separated by the ||| sign ) and includes the formula

The whole formatted row is now:

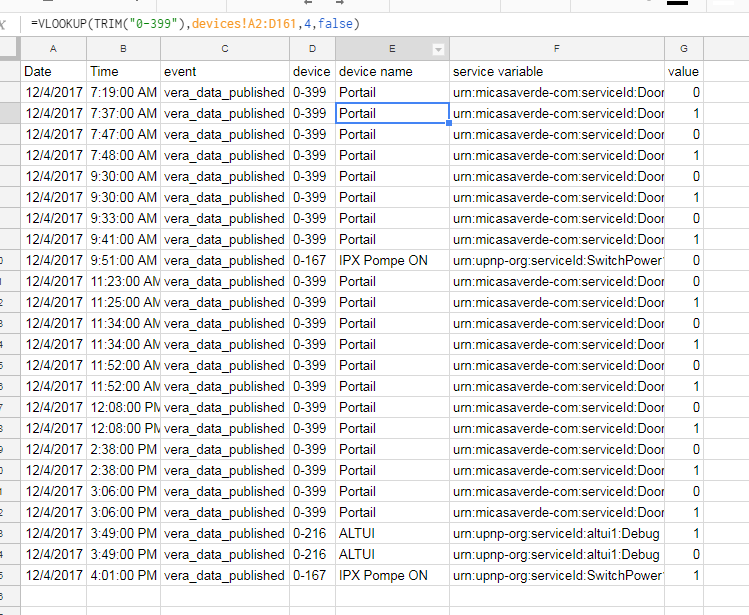
* =DATEVALUE(REGEXREPLACE( "{{OccurredAt}}"," at .\*","")) ||| =TIMEVALUE(REGEXREPLACE( "{{OccurredAt}}",".\* at ","")) ||| {{EventName}} ||| {{Value1}} ||| =VLOOKUP(TRIM("{{Value1}}");devices!A2:D161;4;false)||| {{Value2}} ||| {{Value3}}

Which will produce these fields

1. A date
2. A time
3. The webhook event name
4. The altuiid of the device sending the event
5. The name of that device
6. The “service:variable” name
7. The new value taken by the variable



For a result like this



# CONCLUSION

This is just one example of what can be accomplished with ALTUI IFTTT integration. The possibilities of IFTTT are enormous so the imagination is the limit. Have fun with it and feel free to report back interesting results