Developer: Christopher McNeely
Source: https://github.com/professoraire/ElegantIFTTT-Crestron

Elegant IFTTT Analog Value Parser v1.0

GENERAL INFORMATION

SIMPL WINDOWS NAME	Elegant IFTTT Analog Value Parser v1.0
CATEGORY	Elegant Modules
VERSION	1.0
SUMMARY	Provides a method to parse analog values from messages sent to the Crestron processor from IFTTT.
GENERAL NOTES	This module requires that an Elegant IFTTT Manager module be added to the program. This will handle parsing data received from an event with a matching EventName.
CRESTRON HARDWARE REQUIRED	3-Series Processor

PARAMETERS

ManagerID	S	The ID of the Elegant IFTTT Manager module to associate this module with.
EventName	S	The name of the event to parse information from. This module will only examine the received data if this event name matches the name received from the IFTTT service.
AnalogType	D	The type of analog value to parse. The options are: [0] Raw Integer [1] Percent [2] Signed Integer In the case of Raw Integer, the raw unsigned data is returned to the program. Signed integers are returned as signed values. Percents allow only 0 to 100 as incoming data. (Values larger than 100 are clamped to 100%.) This value is automatically multiplied in Simpl# to provide a full-scale analog percent output.
StringToParse (1-100)	S	A string value to look for in the incoming data. If the desire is to just parse a value, without needing a matching string (for events with only one possible output), then simply enter empty quotes "" in the parameter field. Can use the following prefix qualifiers: • No prefix or = ○ Without any prefix, or with an = sign, the string has to match the entirety of at least one of the data elements. • !

		○ Placing an ! before the string will pulse the output if the string isn’t equal to at least one of the data elements. ● + ○ Placing a + before the string will pulse the output if the string is found anywhere in any of the data elements. This is partial matching and is like doing a Find operation in Simpl+. ● - ○ Placing a – before the string will pulse the output if the string can’t be found anywhere within any of the data elements. This is a partial matching operation. ● >, <, >=, <= ○ Placing a standard mathematical comparison operator before the string will cause the module to parse the string as if it is a value and compare the value of the string to the value of the incoming data elements. The equations used are: StringToParse > DataElement StringToParse < DataElement StringToParse >= DataElement StringToParse <= DataElement The outputs are pulsed if the resulting equation evaluates to true.

CONTROL

Enable	D	When held high this allows the module to parse data. When low the module won’t parse data, even if the event name matches.

FEEDBACK

NewValue (1-100)	D	If a StringToParse element matches the FIRST data element returned from the event (not including the Event Name), then the SECOND data element is evaluated as an analog value and output on the corresponding NewValue analog.

ADDITIONAL DATA

Revision History	v1.0 – Initial Release
Additional Details	This module can fire multiple outputs for each event received from the IFTTT service. Rather than just evaluate for exactly matching string expressions, this module is also capable of checking for strings in the middle of data elements, non-existing strings, or evaluating the data as numerical and performing mathematical comparisons on them as well. The details of how to specify these are provided under the StringToParse parameter notes.

If 100 outputs aren’t sufficient for the quantity of string elements that need to be checked, you can place an additional String Matching Module in the program. Every Elegant IFTTT parsing module will evaluate event data when the Event’s name matches their EventName parameter.

Notes on event strings:
In the IFTTT THAT event, assume that a string is provided like:
Lights|{{LightName}}|{{BrightnessValue}}
In this case, the StringToParse parameter could be given something like: “=Living Room Lamp” and when that string comes in, the value provided is then output the NewValue output. This would allow a user to say “Set the Living Room Lamp to 45%” and have that value output from the module.

If you have an event that always feeds a single value, the StringToParse parameter can be left empty: “”. In this case assume that an IFTTT THAT event provides the string:
Lights|{{BrightnessValue}}
In this case, if the StringToParse parameter is a pair of quotes, then the value would always be output on this analog. Note that it would also be legal to provide an IFTTT THAT event with the string:
Lights||{{BrightnessValue}} (Note the double pipes.) This simply boils down to syntax, as it could be possible to devise an IFTTT THAT event with a key phrase where sometimes you have a named value and sometimes not. The Simpl# code handles checking for both cases.