Ejemplo thingsBoard (TB) – Introducción al Rule Engine

Para profundizar ver: <https://thingsboard.io/docs/user-guide/rule-engine-2-0/re-getting-started/>

## Escenario de uso

Este ejemplo simula el manejo de la luz de un ambiente en base a un sensor LDR y una lámpara de luz con dimer simulada con un Led.

Se muestra:

* cómo enviar telemetría de un LDR hacia al servidor de thingsBoard (livedemo server).
* Cómo procesar los datos utilizando el motor de reglas para controlar la intensidad de un LED en base a paramentos definidos por un usuario
* Cómo enviar un comando al dispositivo para que regule la intensidad del Led.

Los parámetros que se podrán configurar son:

Rangos de luz (en base al valor que devuelve el LDR). En el ejemplo son: darkMin, darkMax, normalMin, normalMax, brightMin, brightMax.

Comandos que se enviaran con el valor del ciclo de PWM que regulará la intensidad del LED. En el ejemplo son: lightDIMCmdValue, lightONCmdValue, lightOFFCmdValue

## La placa y los sensores

La placa nodeMCU tiene los siguientes sensores

* Un LDR con su resistencia (10kOhm) conectado al pin A0
* Un LED con su resistencia (200 Ohm) conectado al pin D0

El nodeMCU necesita conectarse a una red WiFi (por ejemplo, compartida por el celular)

Contar con una cuenta en thingsboard.io live demo server.

Antes de comenzar descargar el sketch y modificar los datos de autenticación para la conexión.

## Pasos para conectarse

* Crear un dispositivo en thingsboard. En este ejemplo tiene el nombre es “NODEMCUGMRULES”

Modificar en el sketch las siguientes sentencias con el nombre del dispositivo y el token de acceso

#define NODE\_NAME "NOMBRE DISPOSITIVO"

#define NODE\_TOKEN "TOKEN DISPOSITIVO"

* Modificar en el sketch las siguientes sentencias con los datos de la red wifi

#define WIFI\_AP "SSID RED"

#define WIFI\_PASSWORD "PASSWORD RED"

Cargar el Sketch al nodeMCU y comprbar con el serial monitor que se conecte a la red y al servidor de TB.

## El código del sketch

El sketch es similar al ejemplo anterior ThingsBoardPaso a Paso.

**En el método setup()** se realiza la conección al servidor y se indica el método de callback para recibir mensajes desde el tópico request (on\_message()).

**En el loop()** se llama a un método **getAndSendTelemetryData()**; que lee el valor del LDR y lo envía por el tópico telemetry mediante un JSON con el siguiente formato

{“valorLDR":"; payload }";

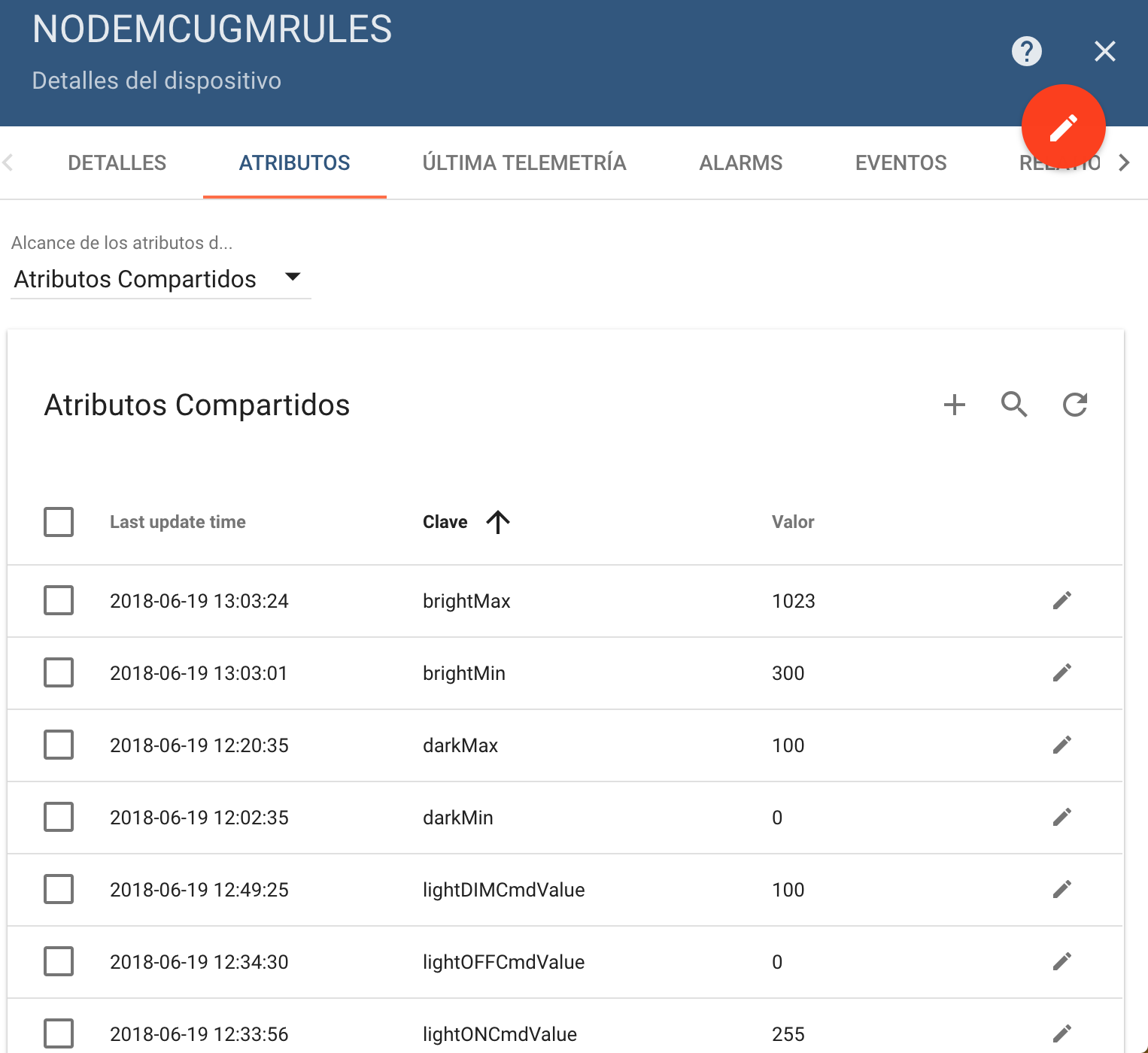
**En on\_message()** se procesas los comandos recibidos desde el servidor. En este caso se recibe un JSON con el nombre del método y su parámetro y se llama al método **void setLightIntensity(int value)**

## Configurar los parámetros para controlar la luz

Para configurar los parámetros de luz y comandos utilizaremos los “Atributos compartidos” que permite definir el dispositivo de Thingboards

En el dispositivo creado en TB, en este caso (NODEMCUGMRULES), en la solapa de atributos seleccionar del Dropbox “Atributos compartidos” y definir los atributos con los valores que se quieren configurar.

Estos atributos permitirán variar el comportamiento del nodeMCU en tiempo de ejecución cambiando los valores de los parámetros.



En este caso definimos los siguientes atributos

|  |  |  |
| --- | --- | --- |
| **Atributo de la luz** | **Tipo** | **uso** |
| brightMin, brightMax | int | Definen el rango que de valores enviados por el LDR que indican un salón con luz natural suficiente para **no** tener que prender la lámpara. **El led se apaga** |
| normalMin, normalMax | int | Definen el rango que de valores enviados por el LDR que indican un salón con luz insuficiente y se tiene que prender la lámpara. **El led se prender en modo dim** |
| darkMin, darkMax | int | Definen el rango que de valores enviados por el LDR que indican un salón sin luz y se tiene que prender la lámpara al máximo. **El led se prender a todo brillo** |

|  |  |  |
| --- | --- | --- |
| **Atributo de Comando** | **Tipo** | **uso** |
| lightDIMCmdValue | int | Indica el valor del ciclo PWM que se utilizara para poner el LED en dim |
| lightOFFCmdValue | int | Indica el valor del ciclo PWM que se utilizara para apagar el LED |
| lightONCmdValue | int | Indica el valor del ciclo PWM que se utilizara para prender el LED |

## Motor de reglas

En TB se pueden definir reglas que procesan los mensajes que se encuentran en los distintos tópicos utilizados en la solución.

En este caso procesaremos los datos enviados por el NodeMCU al tópico de **telemetry** y enviaremos los comandos al NodeMCU por el tópico **request**.

Ambos tópicos se encuentran definidos en la sección correspondiente del **sketch**.

/\*definir topicos.

\* telemetry - para enviar datos de los sensores

\* request - para recibir una solicitud y enviar datos

\* attributes - para recibir comandos en baes a atributtos shared definidos en el dispositivo

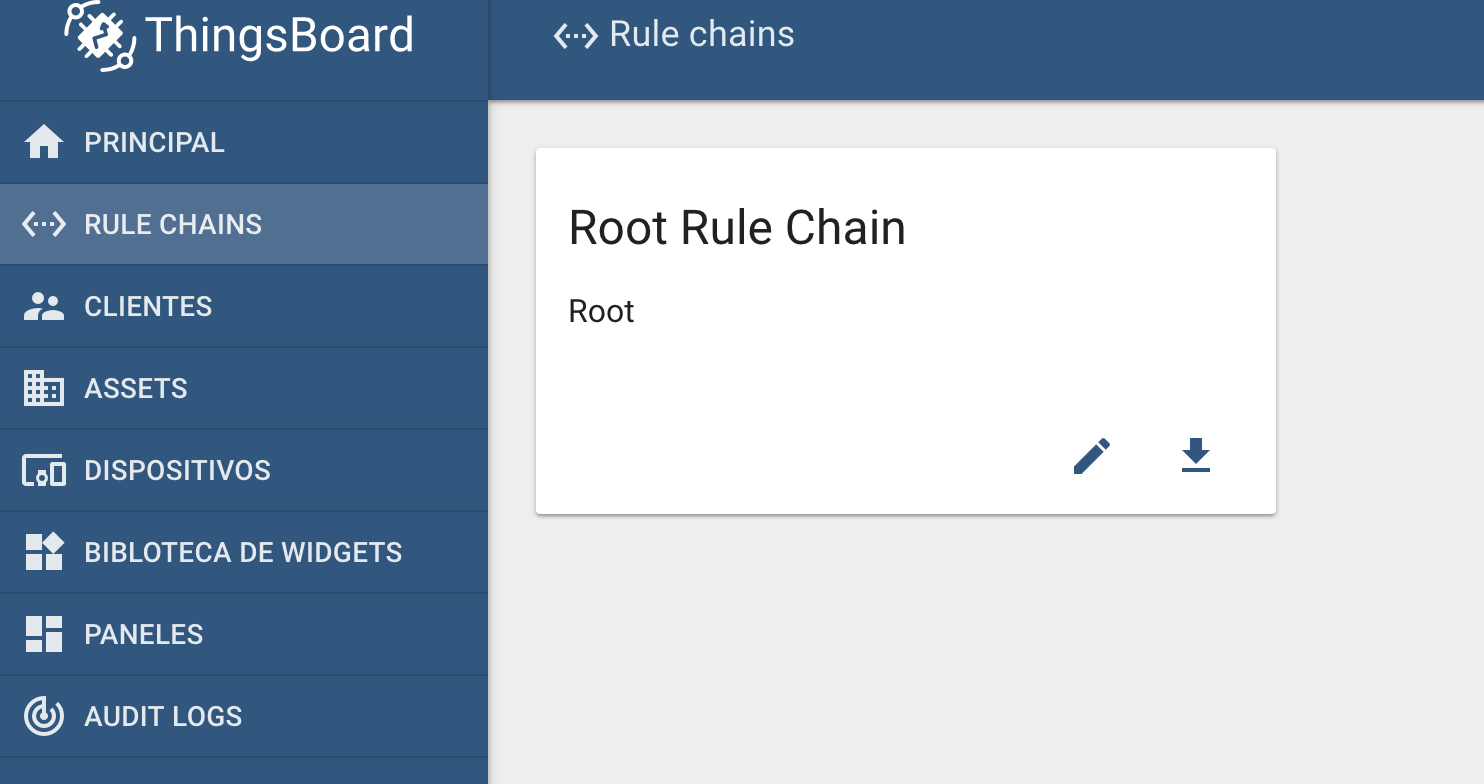
\*/

char telemetryTopic[] = "v1/devices/me/telemetry";

char requestTopic[] = "v1/devices/me/rpc/request/+";

char attributesTopic[] = "v1/devices/me/attributes";

Para procesar los mensajes es necesario definir reglas, esto se hace en “Rule Chains” de TB. En este caso vamos a modificar el “Root rule Chain”que viene por defecto en la demo de TB.



Para abrir la cadena simplemente se clickea sobre la misma

El funcionamiento de la cadena de reglas se basa en encadenar distintos tipos de nodos que procesan los mensajes que se encuentran en los tópicos. Hay nodos para:

* Filtrar mensajes según algún criterio como su tipo, origen, etc.
* Enriquecer los datos de los mensajes utilizando otras variables, Transformar los datos del mensaje por ejemplo agregando nuevos datos
* Transformar los mensajes, por ejemplo, creando un nuevo mensaje o crenado un mail
* Acciones - para enviar mensajes o ejecutar acciones
* Etc.

Los nodos se conectan mediante conectores que se etiquetan según el resultado del procesamiento que realiza el nodo, por ejemplo TRUE, FALSE, SUCCESS, ETC.

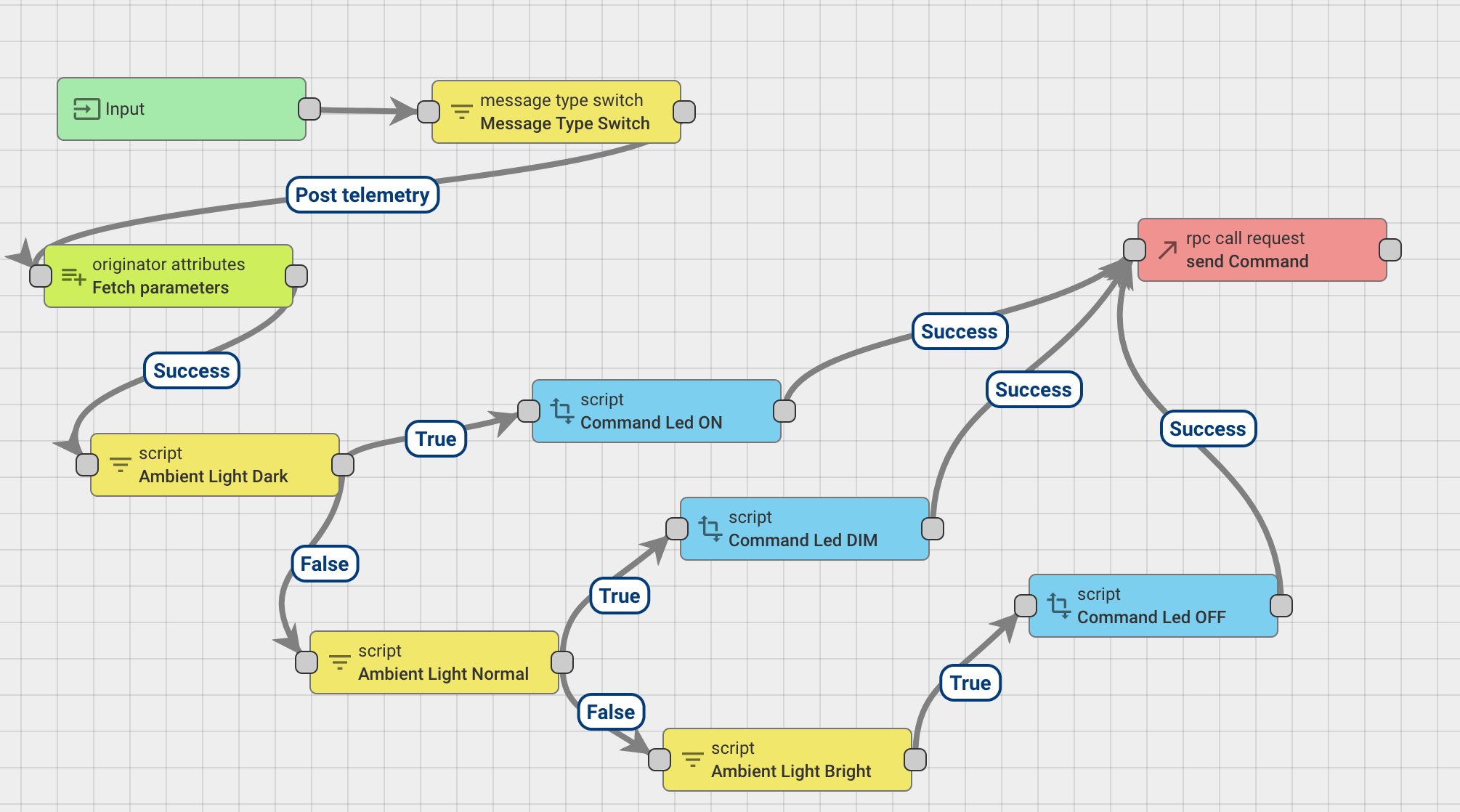
**NOTA**: tener en cuenta que en cada categoría de nodos hay distintos tipos de nodos para script (Filter script, Enrichment script, etc). Su funcionamiento es distinto. En este documento se marca en rojo el tipo de script utilizado para evitar confusiones.

A su vez los mensajes que se transportan por los tópicos tienen la siguiente información:

* **Message** = datos enviados por el tópico, por ejemplo en el de telemetry de este ejemplo el JSON que tiene el valor {“valorLDR":"; payload }";
* **Metadata =** datos sobre el dispositivo, el timestamp (ts) y otros parámetros que nosotros podemos agregar. En este caso se usaran los parámetros de luz y comandos a enviar al nodeMCU.
* **messageType = no se usa**

Por mas detalle sobre los tipos de nodos ver <https://thingsboard.io/docs/user-guide/rule-engine-2-0/filter-nodes/>

A continuación, se muestra la cadena de nodos y sus responsabilidades



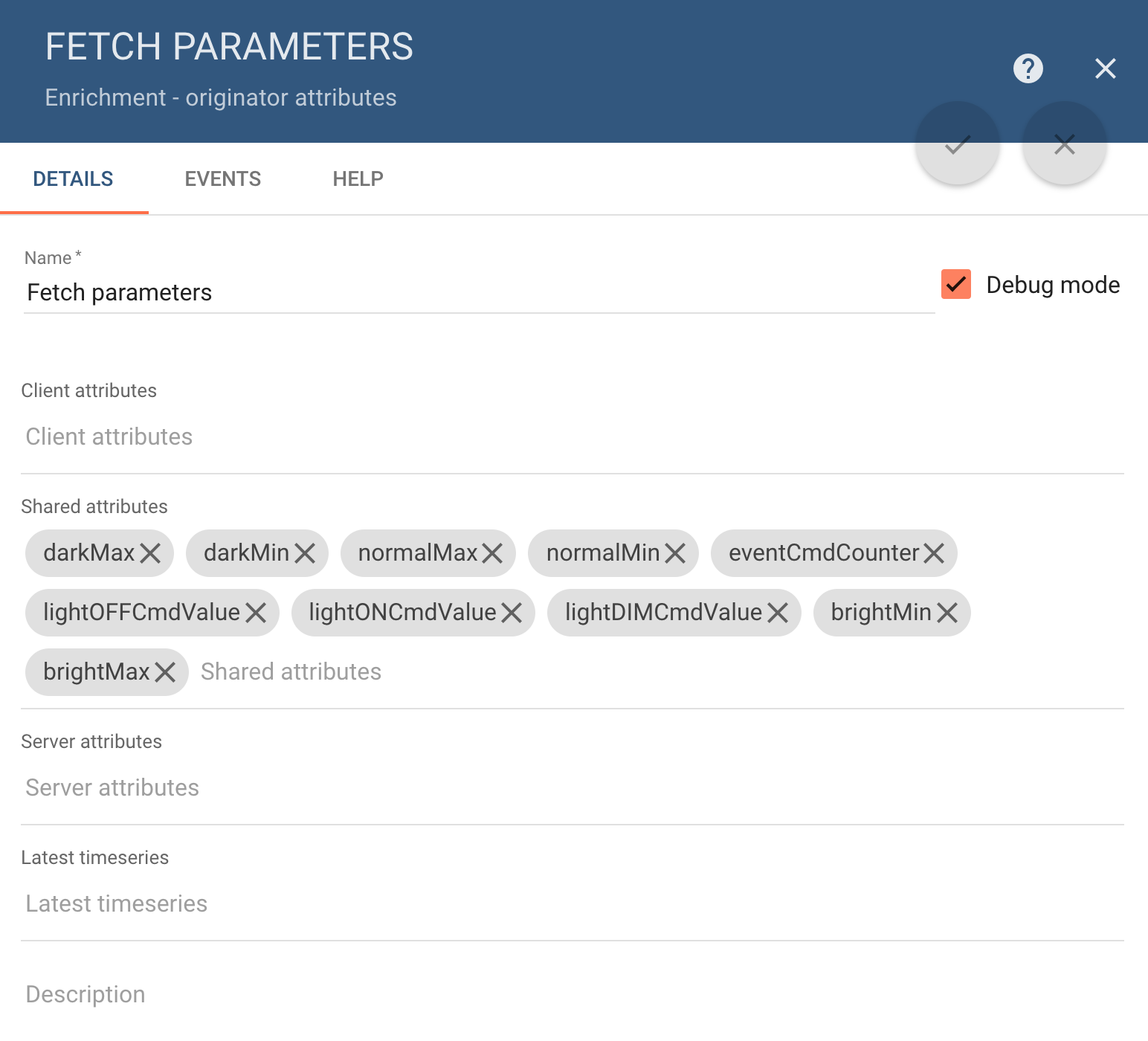
Donde:

**Input** – es un nodo por defecto y que recibe la información que envían los dispositivos por los distintos tópicos.

**Message Type switch (tipo de nodo - Message Type switch)** – filtra los mensajes recibidos. Este nodo no debe cambiarse.

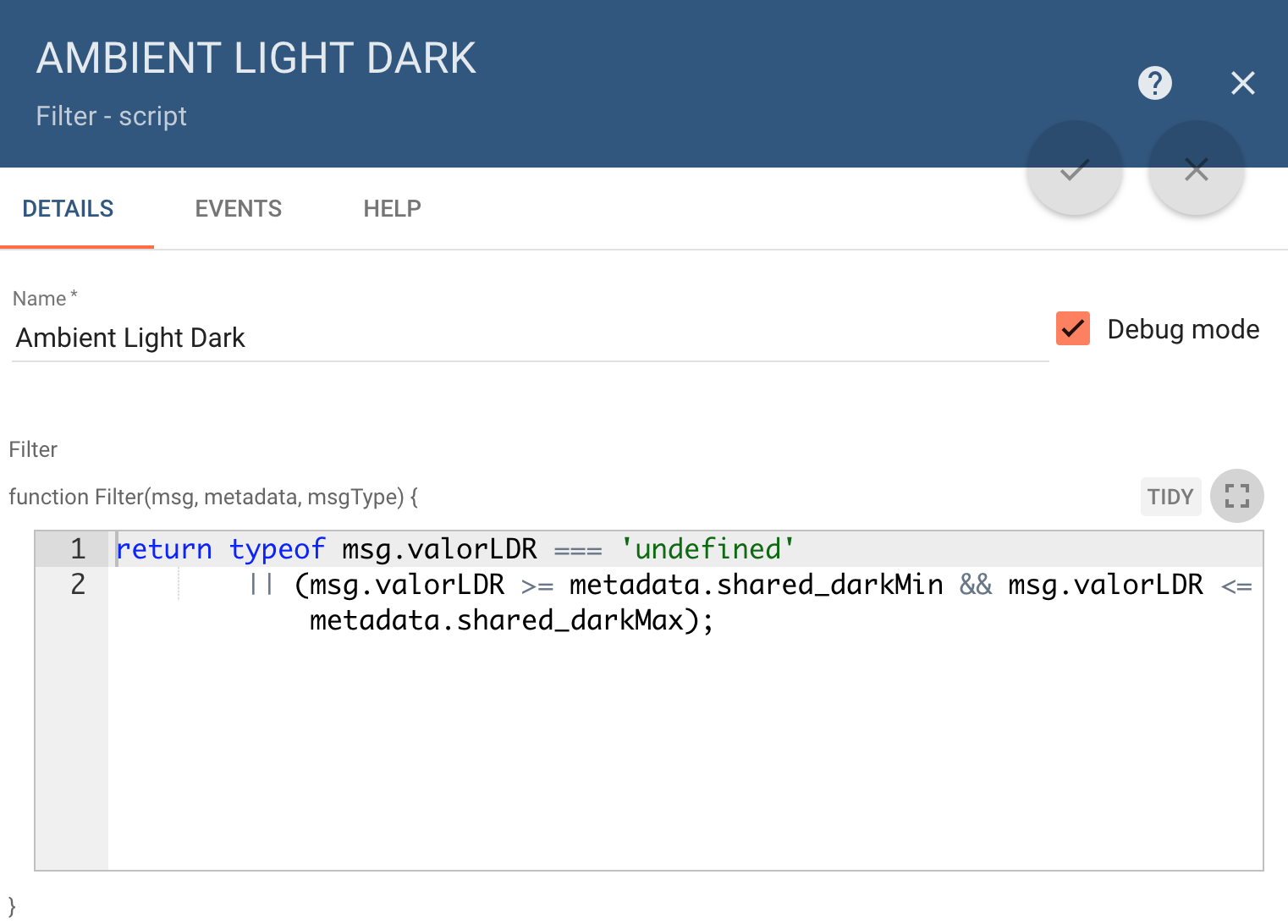
**FETCH PARAMETERS (tipo de nodo - orginator attributes)** – **Notar que este nodo se conecta con el anterior mediante el link (POST TELEMETRY)**, esto es importante porque indica que solo se le pasen los mensajes provenientes de este tópico.

Este nodo lee los atributos compartidos que definen los parámetros (rangos de valores enviados por el LDR y comandos a enviar al nodeMCU). Como se puede ver en la siguiente figura en el campo shared attributes se incluyen todos los parámetros definidos anteriormente en el dispositivo de **TB NODEMCUGMRULES**



Este nodo agrega los valores de estos parámetros en los metadatos de los mensajes que se utilizarán en el siguiente nodo

**AMBIENT LIGHT DARK (tipo de nodo – Filter Script)** – este nodo que se conecta con el anterior mediante el link SUCCESS verifica que los valores enviados por el nodeMCU correspondientes al LDR estén dentro de los parametros definidos para oscuridad. Esto se evalúa en el código JavaScript asociado al nodo



Donde **msg.valorLDR es lo enviado en el JSON por el nodeMCU y metadata.shared\_darkMin y metadata.shared\_darkMax** son los parámetros incluidos en la metadata en el nodo **Fetch Parameters** definido antes.

Este nodo se conecta a sus sucesores mediante los links **TRUE y FALSE.**

En el caso de **FALSE se pasa el mensaje al otro nodo de Filter Script , similar al recién explicado,** queverifica si los valores del LDR están entre el mínimo y máximo para luz normal AMBIENT LIGHT NORMAL

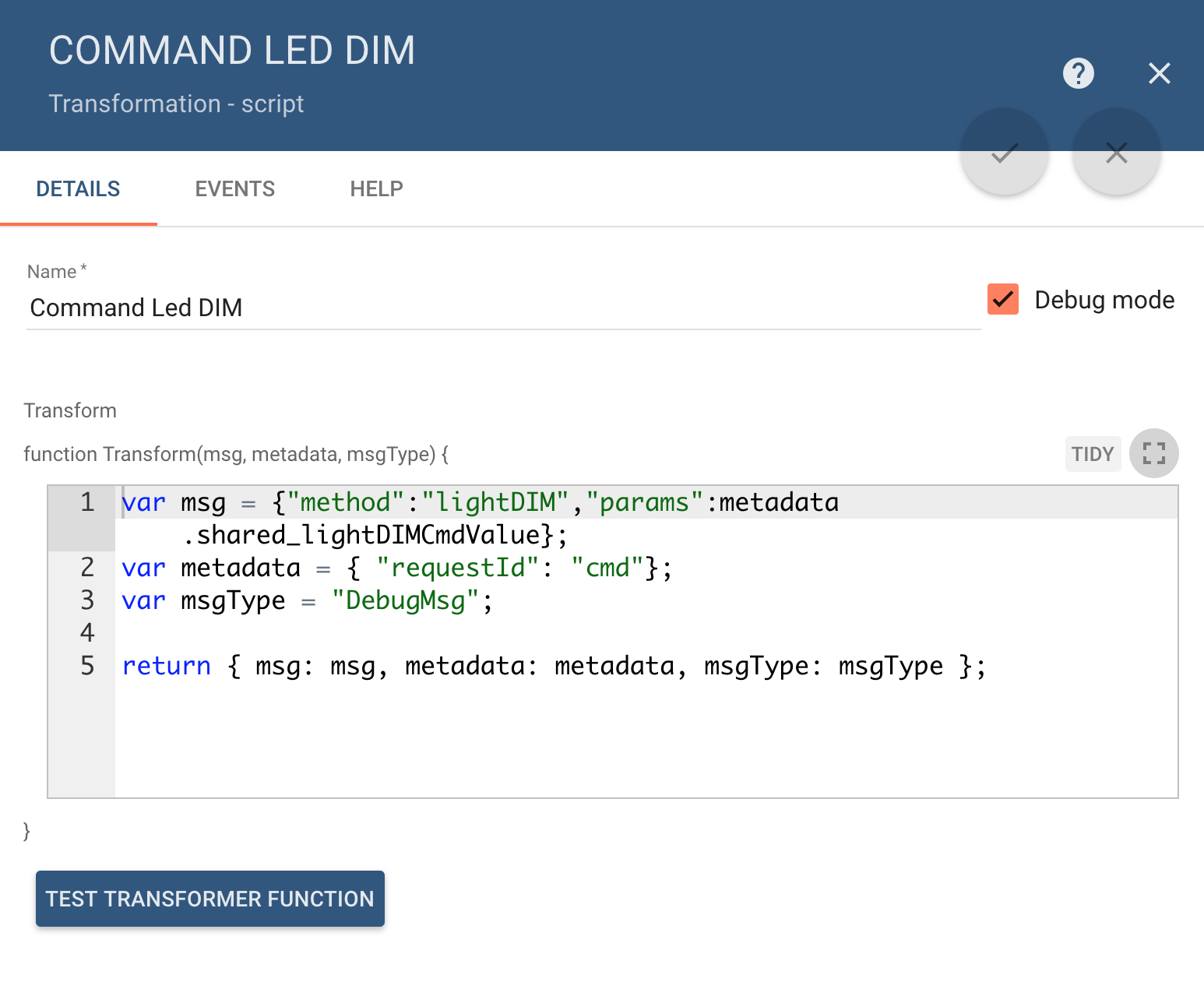
En el caso de **TRUE** se pasa el mensaje a otro nodo (COMMAND LED ON) para que envié el comando al nodeMCU para que prenda la luz.

**COMMAND LED ON** (**tipo de nodo – transformation script)** – este nodo genera el mensaje de comando que se va a enviar al NodeMCU para que prenda la luz.

Como se puede ver en la siguiente figura el código JavaScript genera un JSON con el comando y lo pone en el atributo message (msg).

{"method":"lightDIM","params":metadata.shared\_lightDIMCmdValue};

El método es el que se va a reconocer en el on-message del sketch y el parámetro se obtiene de los metadados que fueron agregados en el Nodo **Fetch Parameters.**



Este nodo se conecta al siguiente mediante un link del tipo **SUCCESS** con el nodo **SEND COMMAND** que se describe a continuación.

**SEND COMMAND** (**tipo de nodo – rpc call request) –** Este nodo envía el comando generado en el nodo anterior al tópico request para que lo reciba en NodeMCU y lo procese mediante el método on\_message () del sketch.

El resto de los nodos de la cadena tienen un comportamiento similar, lo único que varía es la función JavaScript para evaluar el valor del LDR y el comando a enviar.

**Se adjunta al ejemplo un archivo .js con el código de cada uno de los nodos de la cadena.**

**NOTA**: esta cadena de filtros se podría simplificar utilizando en lugar de un nodo para verificar cada rango de valores del LDR (nodos AMBIENT LIGHT \*) un filtro del tipo Filter / switch en el cual se combine el código de cada uno de estos nodos.