webdyn
flexitron group

ROUTER TITAN

Nota de aplicación 78

Uso de un Router Titan para la lectura de un Contador Eléctrico DLMS/COSEM

de forma autónoma con envío de datos leídos a plataforma MQTT

Lectura de un Contador Eléctrico DLMS/COSEM

Detalles del escenario

Los router Titan disponen de todas las funcionalidades típicas de un router 4G/3G/2G pero además cuentan con una serie de prestaciones adicionales que lo convierten en uno de los routers con más prestaciones del mercado. Una de las prestaciones adicionales es la capacidad de leer de forma autónoma contadores con protocolo DLMS/COSEM, pudiendo leer periódicamente los valores instantáneos, almacenando los datos en su datalogger interno para enviarlos a una plataforma (HTTP / HTTPS, MQTT / MQTTS, FTP) siempre que tenga cobertura 4G/3G/2G

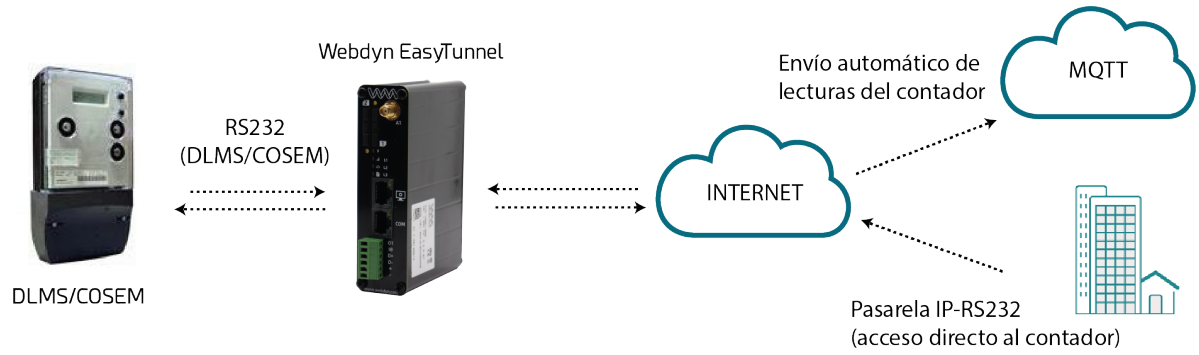
En concreto se pretende lo siguiente: cada minuto, el dispositivo Webdyn EasyTunnel debe leer los registros Modbus con dirección 40000 y 40001 del PLC1 y los registros 40000 y 40001 del PLC2. En ambos PLCs el registro 40000 corresponde con la temperatura medida y el registro 40001 con la humedad. Las lecturas realizadas deben almacenarse en la memoria interna no volátil del Webdyn EasyTunnel (en su datalogger) y éste debe enviar dichos datos leídos a la plataforma MRI eSight siempre que sea posible (haya cobertura, conectividad IP, ...) La comunicación con los PLCs es a través de un bus RS485 con una configuración 9600,8,N,1.

1. Descripción del escenario de ejemplo

- Se dispone de un Contador Eléctrico ITRON SL7000 con protocolo DLMS/COSEM, con puerto serie RS232 y configuración 9600,8,N,1.

- Se pretende configurar un router Titan (en el caso de este ejemplo se utilizará un modelo Webdyn-Easy-Tunnel, al que se hará referencia como “router Titan”) para leer autónomamente, cada 15 minutos, el contador DLMS/COSEM y extraer ciertos valores instantáneos (Energía Activa Positiva, Energía Activa Negativa, Energía Reactiva Inductiva, Energía Reactiva Capacitiva, Potencia Activa Positiva, Potencia Activa Negativa, Potencia Reactiva Activa, Potencia Reactiva Negativa, Voltaje Fase I, Voltaje Fase II, Voltaje Fase III). Los valores leídos del Contador Eléctrico deberán enviarse a un broker MQTT.

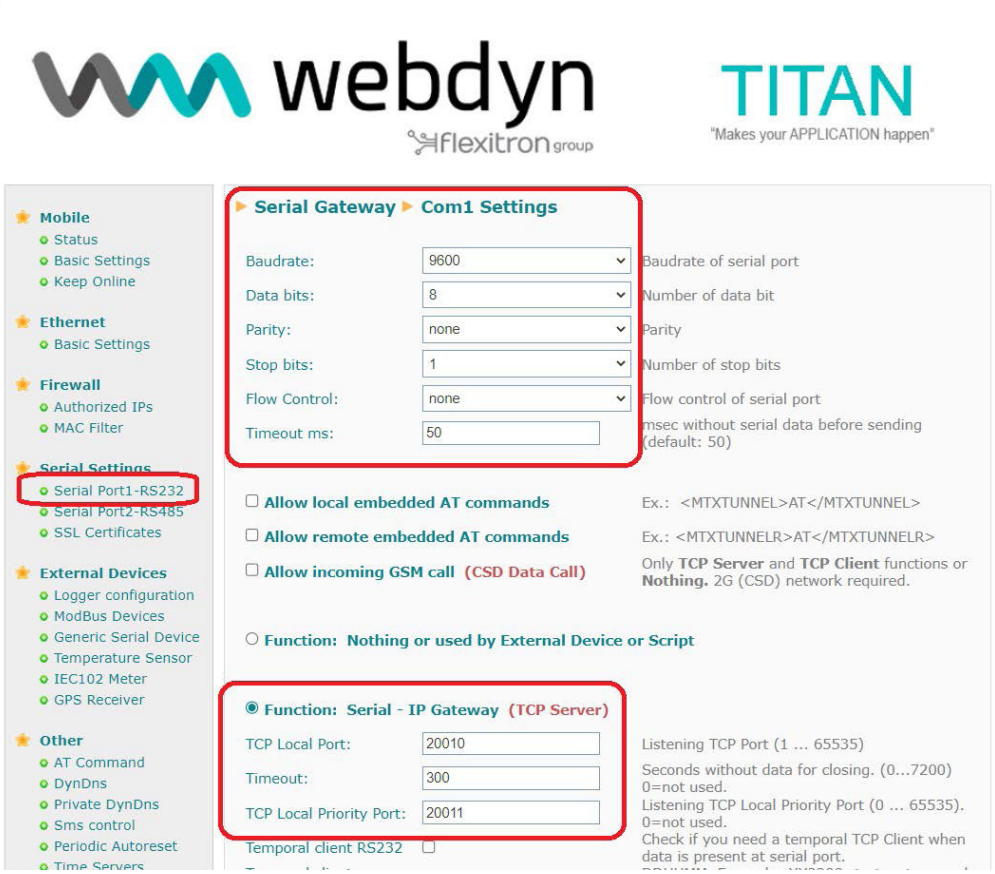
- También se desea tener la capacidad de poder acceder en cualquier momento de forma remota al contador DLMS/COSEM, por lo que el dispositivo Titan también debe poderse comportar como una pasarela IP-RS232 trasnparente sobre el contador. Esta conexión remota ocasional debe tener prioridad sobre la lectura autónoma.



2. Configuración del puerto serie asociado

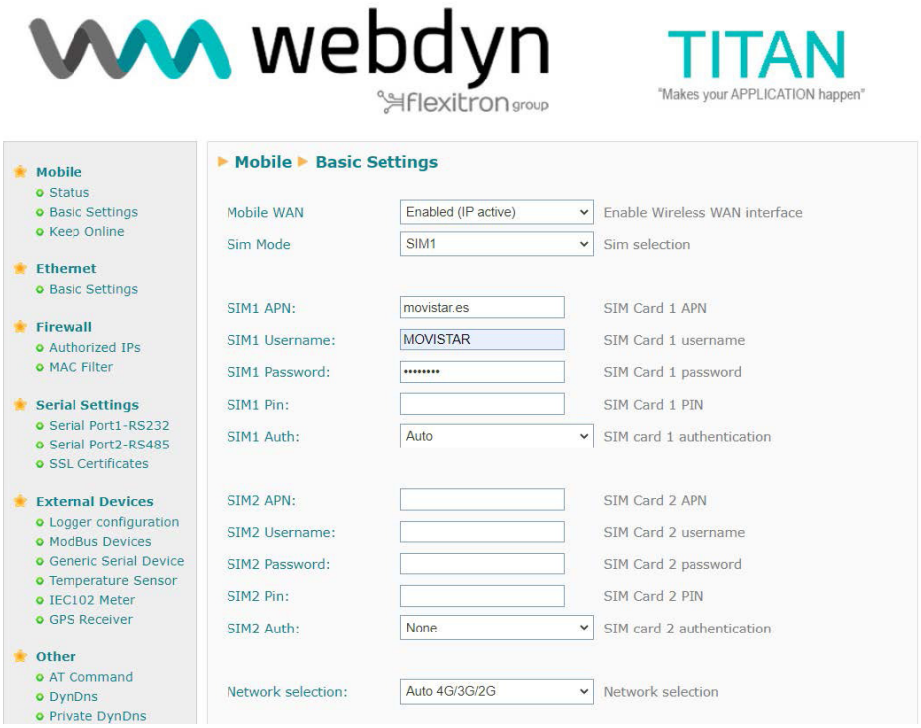
En este ejemplo se utilizará el puerto RS232 del router Titan, pues es el puerto serie que estará conectado al Contador Eléctrico DLMS/COSEM. Para configurar el puerto serie hay que acudir al menú "Serial Settings → Serial Port1-232" y configurar los valores apropiados. Los valores apropiados son aquellos que coincidan con la configuración del puerto serie del Contador Eléctrico que en el caso de este ejemplo son 9600,8,N,1.

Debe configurarse una pasarela TCP Server. Utilizaremos como puerto LOCAL el puerto TCP 20010. Este puerto TCP 20010 será utilizado internamente por el SCRIPT de lectura DLMS/COSEM para la comunicación local con el contador. El puerto TCP 20011 será el puerto TCP que servirá para conectar directamente con el contador de forma remota a través de una pasarela IP-RS232 transparente. Este puerto TCP 20011 será prioritario sobre el puerto TCP 20010, lo que se traduce en que las comunicaciones locales (las del script comunicándose con el contador) se verán suspendidas mientras dure una comunicación remota sobre el puerto TCP 20011 de la pasarela transparente IP-RS232.



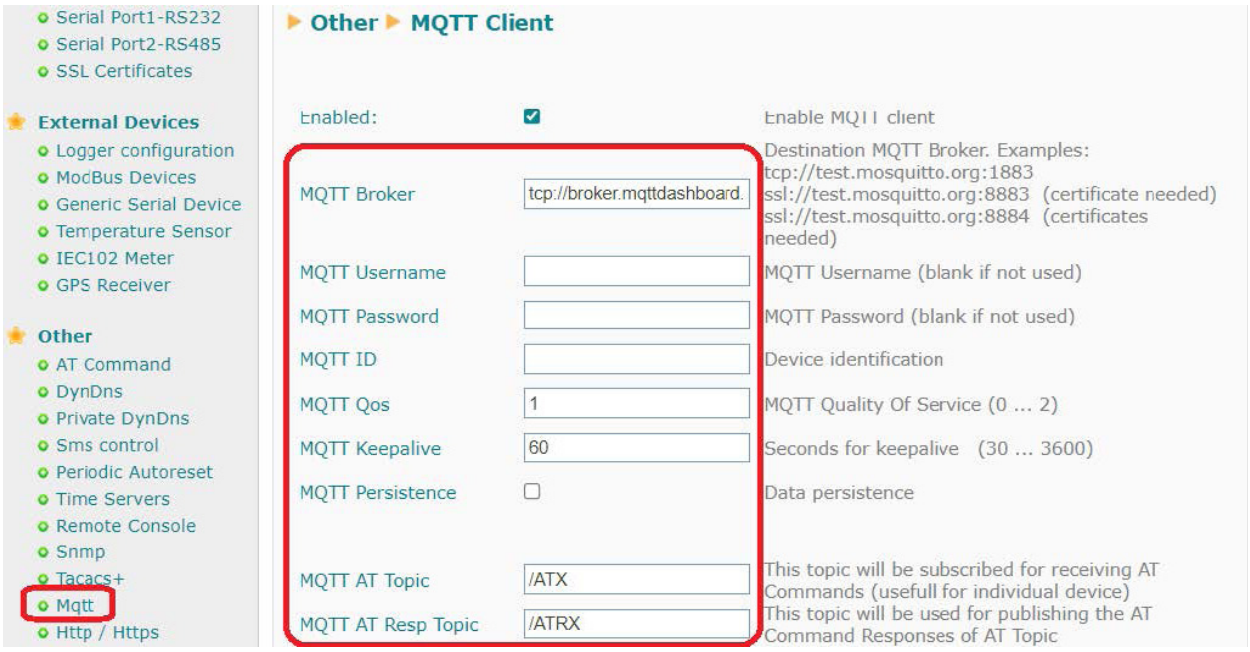
3. Configuración Mobile

El router Titan necesita ser configurado para disponer de una comunicación IP a través de 4G/3G/2G para comunicaciones vía MQTT y para la pasarela remota sobre el puerto TCP 20011. Para ello hay que acudir al menú “Mobile → Basic Settings” y como mínimo se debe modificar la configuración para habilitar la interfaz WAN 4G/3G/2G y especificar el APN / username / password de la tarjeta SIM. Como en este escenario no es preciso la llamada CSD, es posible (y recomendable) especificar en el campo “Network Selection” con el valor “Auto (4G/3G/2G)”



4. Configuración MQTT

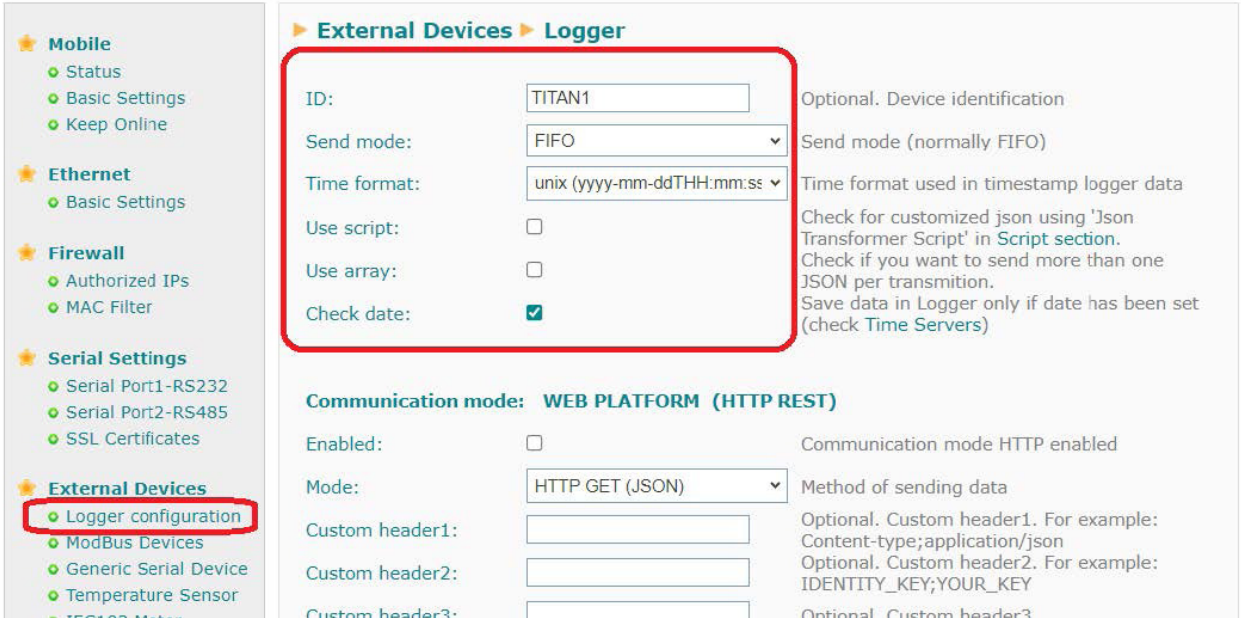
El router Titan enviará periódicamente los datos leídos del Contador Eléctrico DLMS/COSEM a un broker MQTT por lo que debe configurarse también la sección MQTT. Para ello debe acudirse al menú “Other → Mqtt” y establecer la configuración apropiada. En el caso de este ejemplo se va a utilizar la plataforma de pruebas mqtt HIVEMQ. Para ello, en el campo “MQTT Broker”, se debe indicar “tcp://broker.mqttdashboard.com:1883” y en el campo “MQTT ID” usaremos por ejemplo el IMEI del router Titan, indicando “[IMEI]”. También se pretende poder enviar comandos AT al router Titan desde la plataforma MQTT para tareas de mantenimiento, configuración, lectura de estados del router, etc. Para poder enviar remotamente comandos AT al router Titan deben completarse los campos “MQTT AT Topic” y “MQTT AT Resp Topic”. El primer topic es donde se deben enviar los comandos AT para que sean recibidos y ejecutados por el router Titan. El segundo topic es donde el router Titan enviará las respuestas a los comandos AT ejecutados, es decir, el topic con el que podremos obtener la respuesta del comando AT ejecutado. La siguiente captura de pantalla muestra la configuración indicada para este escenario.



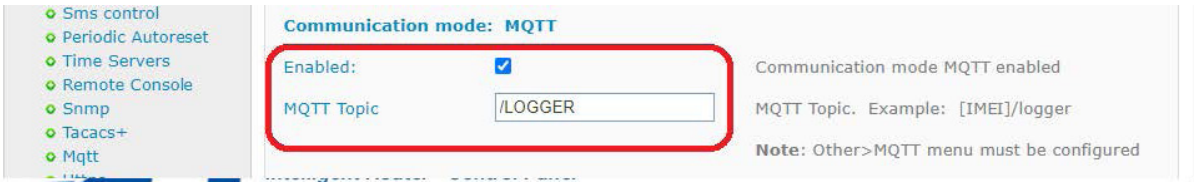
5. Configuración LOGGER

El siguiente paso es configurar el LOGGER. Es decir, se debe configurar el espacio de memoria interna del router Titan donde desde el SCRIPT (que veremos a continuación) guardará los datos leídos del Contador Eléctrico DLMS/COSEM y también debe configurarse el método de envío de dichos datos a la plataforma remota, que en este ejemplo será via MQTT. La configuración del LOGGER se realiza desde el menú “Extenal Devices → Logger configuration”.

En esta sección puede configurarse el campo opcional ID. En este ejemplo será el texto “TITAN1”. Escogeremos el método de envío “FIFO” para el envío de datos y el formato de hora “unix”. También seleccionaremos la casilla “Check date” para que sólo se almacenen datos en el Logger si la fecha es correcta, ya que los datos que se envíen a la plataforma deben tener un timestamp correcto (por lo que también deberá configurarse correctamente la sección “Other → Time Servers”)

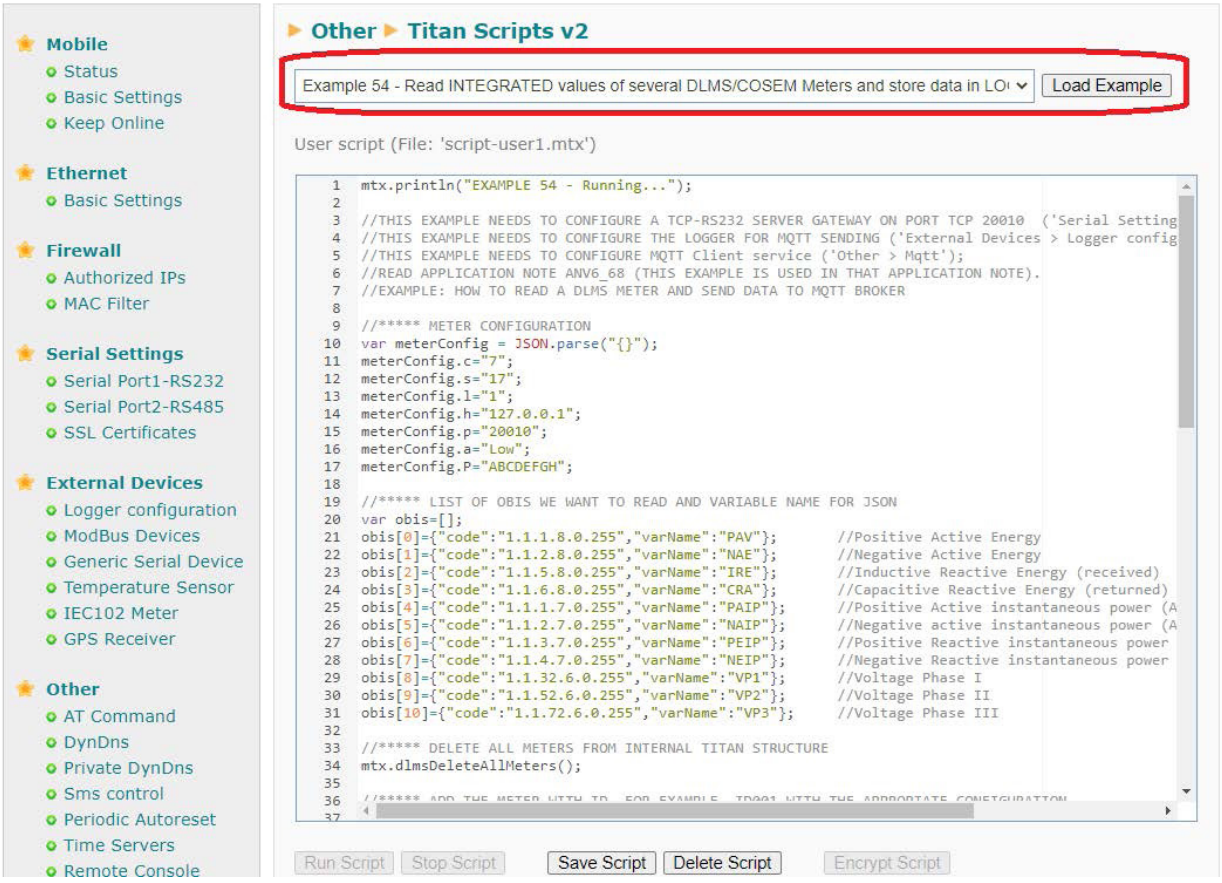


En esta misma sección de configuración del LOGGER, en la parte inferior de la pantalla, debe seleccionarse la casilla “Enabled” para activar el modo de envío por MQTT. Como topic de envío (el topic MQTT que el router Titan utilizará para enviar los datos leídos del contador) indicaremos por ejemplo el texto “/LOGGER”.



6. Configuración del SCRIPT

El último paso es programar el script que realice la lectura del contador DLMS/COSEM, componga un objeto JSON con los datos leídos del contador y los almacene en la memoria del LOGGER para su posterior envío automático a la plataforma MQTT. Para programar este script debe acudirse a la sección “Other → Titan Scripts”. En este caso no será necesario escribir manualmente todo el código del ejemplo, pues éste forma parte del ejemplo 54 (titulado “Read DLMS/COSEM meter and store data in LOGGER”). Es decir, basta con seleccionar el ejemplo del menú desplegable y pulsar el botón “Load Example” para cargarlo en memoria y sólo modificar lo pertinente.



A continuación se describirá el código del ejemplo propuesto. El siguiente fragmento de código define el JSON de configuración del contador. Esta configuración es necesaria pasarla como parámetro de la función mtx.dlmsAddMeter() como se verá más adelante.

```
//***** METER CONFIGURATION
var meterConfig = JSON.parse("{}");
meterConfig.c="7";              //Client Address
meterConfig.s="17";             //Server Address
meterConfig.l="1";              //Logical Server Address
meterConfig.h="127.0.0.1";      //Local IP of TCP-RS232 local gateway
meterConfig.p="20010";          //Local TCP port of TCP-RS232 local gateway
meterConfig.a="Low";            //Authentication
meterConfig.P="ABCDEFGH";       //Password for Authentication

```

En el siguiente fragmento de código, en los parámetros “code” del array de json “obis”, se indican los códigos OBIS que se pretenden leer del contador DLMS/COSEM. En el parámetro “varName” indicamos el nombre del parámetro que utilizaremos después a la hora de crear el JSON con los datos leídos.

```
//***** LIST OF OBIS WE WANT TO READ AND VARIABLE NAME FOR JSON
var obis=[];
obis[0]={"code":"1.1.1.8.0.255","varName":"PAV"};       //Positive Active Energy
obis[1]={"code":"1.1.2.8.0.255","varName":"NAE"};       //Negative Active Energy
obis[2]={"code":"1.1.5.8.0.255","varName":"IRE"};       //Inductive Reactive Energy (received)
obis[3]={"code":"1.1.6.8.0.255","varName":"CRA"};       //Capacitive Reactive Energy (returned)
obis[4]={"code":"1.1.1.7.0.255","varName":"PAIP"};      //Positive Active instantaneous power (A
obis[5]={"code":"1.1.2.7.0.255","varName":"NAIP"};      //Negative active instantaneous power (A
obis[6]={"code":"1.1.3.7.0.255","varName":"PEIP"};      //Positive Reactive instantaneous power
obis[7]={"code":"1.1.4.7.0.255","varName":"NEIP"};      //Negative Reactive instantaneous power
obis[8]={"code":"1.1.32.6.0.255","varName":"VP1"};      //Voltage Phase I
obis[9]={"code":"1.1.52.6.0.255","varName":"VP2"};      //Voltage Phase II
obis[10]={"code":"1.1.72.6.0.255","varName":"VP3"};     //Voltage Phase III
```

En el siguiente fragmento de script se borran todos los contadores que hubieran en la estructura inicialmente y añadimos el único contador que se va a leer en este ejemplo al que adjudicamos como identificador “ID001” como primer parámetro. Como segundo parámetro de la función pasamos la configuración de lectura del contador que comentamos anteriormente y que tenemos almacenada en la variable meterConfig.

```
//***** DELETE ALL METERS FROM INTERNAL TITAN STRUCTURE
mtx.dlmsDeleteAllMeters();

//***** ADD THE METER WITH ID, FOR EXAMPLE, ID001 WITH THE APPROPIATE CONFIGURATION
mtx.dlmsAddMeter("ID001",JSON.stringify(meterConfig));
```

En el siguiente fragmento de código basicamente se realiza de forma cíclica y periódica la conexión con el contador DLMS/COSEM con la función mtx.dlmsConnect(). Si la conexión es correcta se realiza la lectura del valor de cada registro OBIS (concretamente el índice 2, donde se encuentra el valor) mediante la función mtx.dlmsGetAttributeValue(). Una vez leído el valor del registro OBIS, vamos creando el objeto JSON con los datos leídos. Finalmente, leídos todos los registros OBIS, el json se guarda en

la memoria del Logger (que enviará automáticamente los datos vía MQTT) mediante la función mtx. loggerWrite(). Tras ello se realiza la desconexión del contador mediante la función mtx.dlmsDisconnect(). Debe tenerse en cuenta que la primera vez que se lee el contador puede tomar bastante tiempo (~2 minutos) porque se obtiene toda la estructura del mismo. Valores que se guardan en memoria caché, por lo que las siguientes lecturas serán más rápidas.

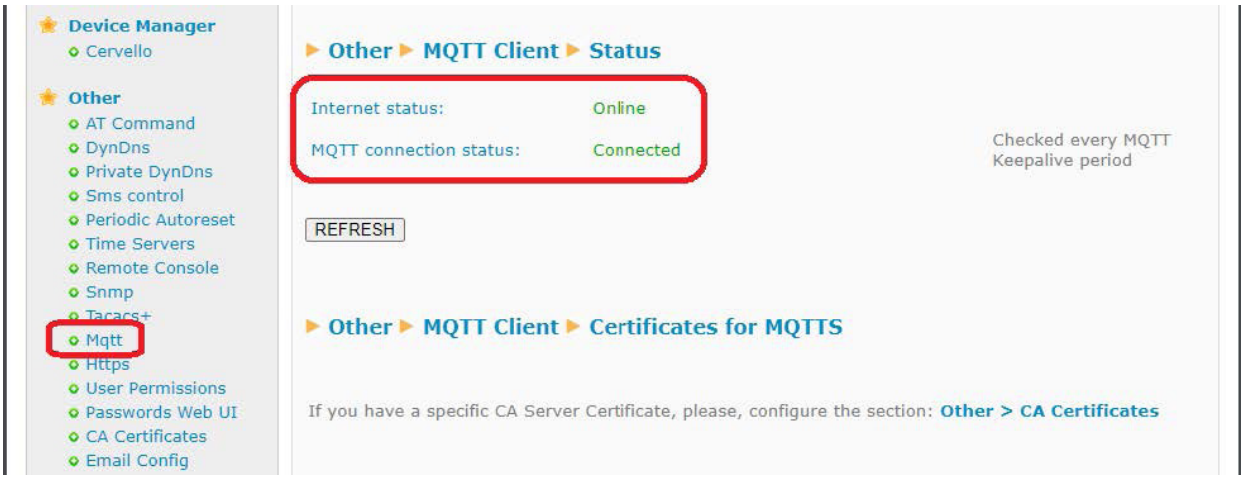
```
{
    //***** CONNECT WITH THE DLMS METER
    var resConnect=mtx.dlmsConnect("ID001");
    if (resConnect)
    {
        var jsonData="{";
        //***** FOR EACH OBIS ...
        for (var i=0;i<obis.length;i++) {
            //***** GET THE VALUE OF OBIS (ATTRIBUTE NUMBER 2)
            var value=mtx.dlmsGetAttributeValue(obis[i].code,2);
            //***** IF READING WAS OK ...
            if (value[0].length>0)  {
                mtx.println("OBIS (" + obis[i].code + "): " + value[0]);
                //***** ADD TO JSON THE VARIABLE WITH ITS  VALUE
                jsonData=jsonData + "\"" + obis[i].varName + "\":" + value[0] + ",";
            }
            //***** IF READING WAS ERROR ...
            else {
                mtx.println("OBIS (" + obis[i].code + "): ERROR: " + value[1]);
                jsonData=jsonData + "\"" + obis[i].varName + "\":\"ERROR\",";
            }
        }
        if (jsonData.endsWith(","))
            jsonData=jsonData.substring(0,jsonData.length()-1);
        jsonData=jsonData + "}";

        //***** WRITE DATA INTO LOGGER FOR SENDING ...
        mtx.loggerWrite(jsonData);

        mtx.println("JSON DATA:" + jsonData);
    }
    mtx.dlmsDisconnect();
```

7.Probando el escenario

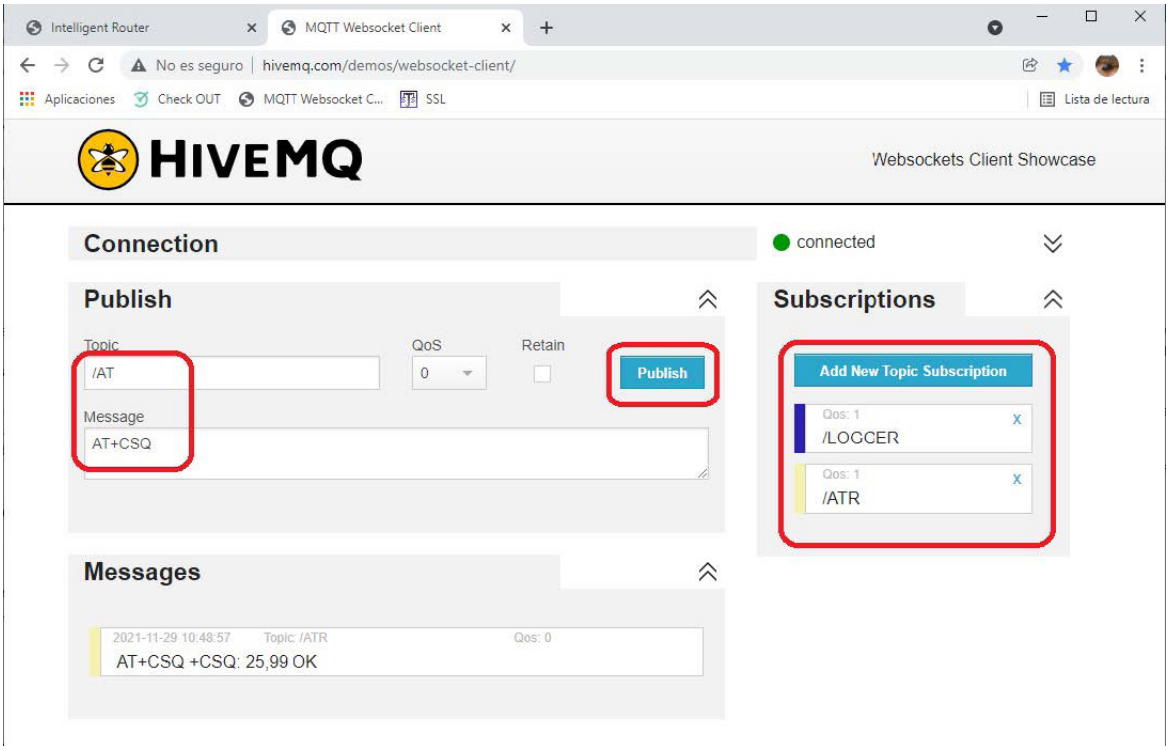
Por último queda comprobar el correcto funcionamiento del sistema. Una vez reiniciado el router Titan y pasados unos segundos debe comprobarse que el router ha obtenido una dirección IP en el menú

“Mobile → Status”

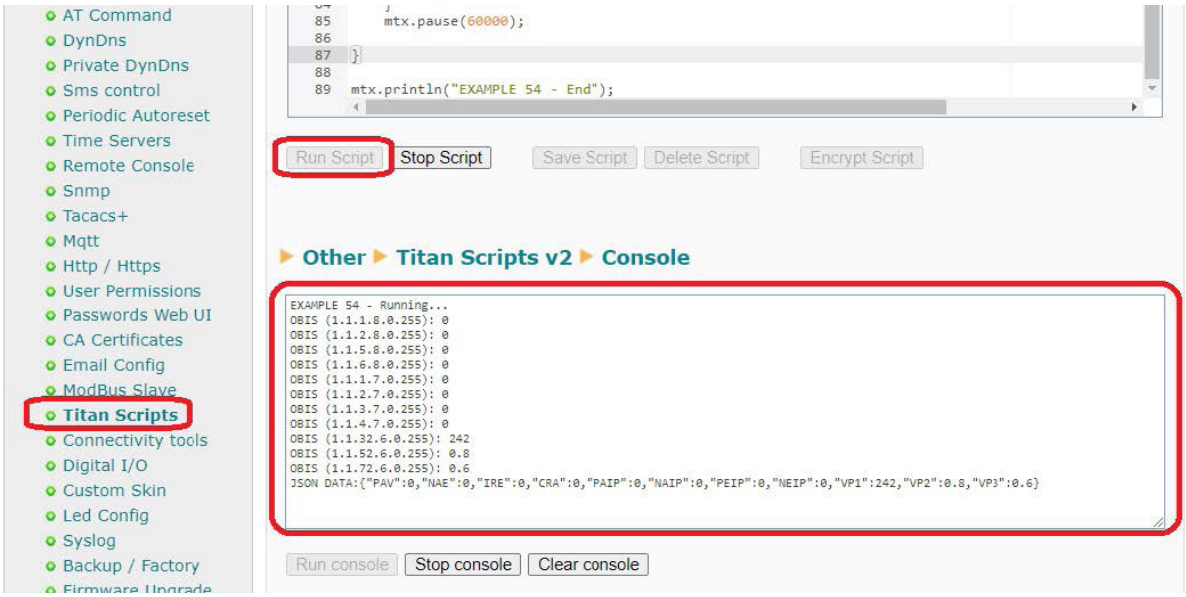
Seguidamente en el menú “Other→Mqtt” deberá comprobarse que la conexión con el broker MQTT sea correcta.



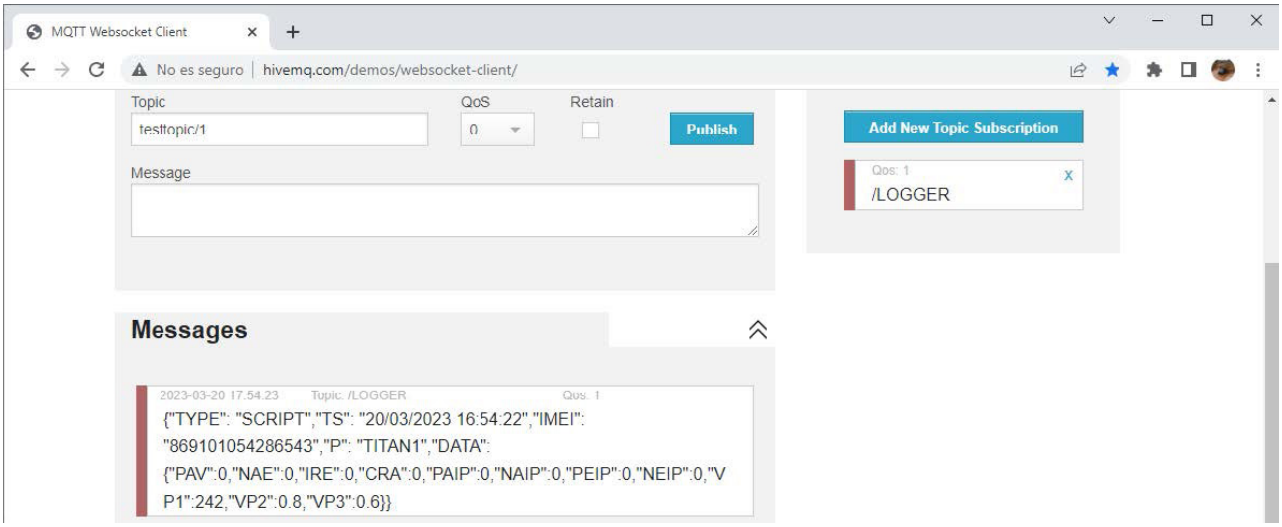
También es posible comprobar en el broker MQTT que la conexión existe y que está bien establecida enviando un comando AT al dispositivo Titan vía MQTT. Para ello deben configurarse en la plataforma los topics “/ATR” (para recibir la respuesta de la ejecución de comandos AT) y “/LOGGER”, donde el rotuer Titan enviará los datos leídos del Contador Eléctrico DLMS/COSEM.



Ya por último vamos a comprobar si la comunicación con el contador DLMS/COSEM es correcta. Para ello podemos ejecutar manualmente el Script pulsando el botón “Run Script” y podremos observar el resultado en la salida de consola. Si todo funciona correctamente se podrá visualizar el valor de cada uno de los registros OBIS leídos y del JSON construido.



Si la lectura del contador es correcta, también deberíamos ver en el broker MQTT que se reciben los datos en el topic configurado anteriormente "/LOGGER", como puede verse a continuación.



¿Más dudas?

Escríbenos tus consultas a iotsupport@mtxm2m.com