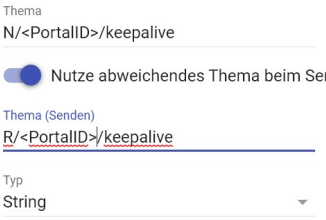
1. Verbindung Cerbo mit IPSymcon herstellen

1.1. MQTT im Cerbo aktivieren

Einstellungen->Dienste MQTT auf LAN (SSL) einschalten

1.2. Client Socket anlegen

1.3. MQTT Client Device und MQTT Client anlegen



MQTT Client auf den erstellten Client Socket verlinken

Keepalive script erstellen und mindestens alle 60 Sekunden ausführen (bei mir alle 30 Sekunden)
target id die VariablenID der Variablen value unter der Instanz. Bei einem leeren String werden alle Werte gesendet, es kann aber auch ein Array mit den zu übertragenden Topics angegeben werden (verringert die Datenmenge)

```
<?php
//RequestAction(22282,'["battery/#","system/#","settings/#","solarcharger/#","temperature/#","multi/#","+/+/ProductId"]')
;
RequestAction(22282,'');
```

2. Variablen in IPSymcon anlegen:

2.1. MQTT-Konfigurator öffnen

Hier sollten nun bereite alle angeforderten oder alle Topics erscheinen.
Benötigte Variablen anlegen (Bei mir erscheinen ca. 4500 Variablen und ich habe 500 davon angelegt)

2.2. Variablen konfigurieren

Profile erstellen und zuordnen, Archiveinstellungen vornehmen, Webfront erstellen

2.3. Variablenprofile anlegen

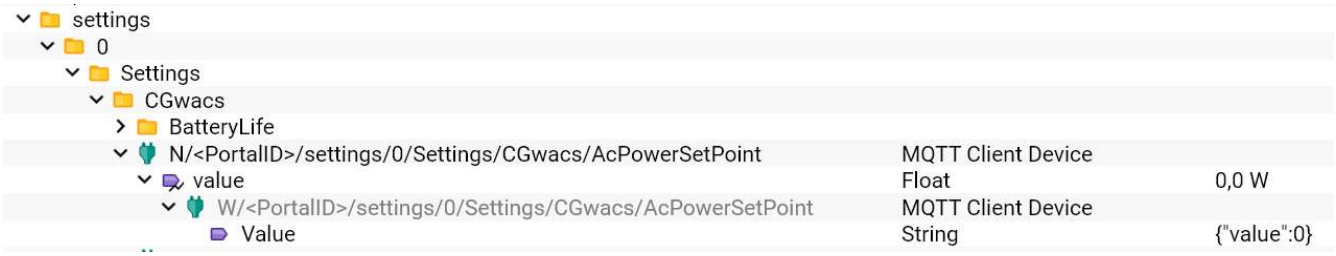
Um nicht alles per Hand anlegen zu müssen kann man sich natürlich für viele Dinge Scripts schreiben.
So habe ich zum Beispiel ein Script, das anschließend für die angelegten Variablen Links anlegt diese in Kategorien einordnet, damit das Webfront bereits halbwegs strukturiert ist.

Als Beispiel hier ein Script zum automatischen anlegen von Variablenprofilen, die relativ viele Assoziationen haben.
Hier dient als Vorlage übrigen das ExcelFile <CCGX-Modbus-TCP-register-list-3.00.xlsx>. Dort ist auch der dbus-obj-path angegeben, der meist dem Topic entspricht.

```
function CreateVariableProfileAssociations() {
//$table mit dem Wert dbus-unit aus dem Excelfile befüllen und das bereits erstellte Profil updaten
$table="0=No error;1=Battery initialization error;2=No batteries connected;3=Unknown battery connected;4=Different
battery type;5=Number of batteries incorrect;6=Lynx Shunt not found;7=Battery measure error;8=Internal calculation
error;9=Batteries in series not ok;10=Number of batteries incorrect;11=Hardware error;12=Watchdog error;13=Over
voltage;14=Under voltage;15=Over temperature;16=Under temperature;17=Hardware fault;18=Standby shutdown;19=Pre-charge
charge error;20=Safety contactor check error;21=Pre-charge discharge error;22=ADC error;23=Slave error;24=Slave
warning;25=Pre-charge error;26=Safety contactor error;27=Over current;28=Slave update failed;29=Slave update
unavailable;30=Calibration data lost;31=Settings invalid;32=BMS cable;33=Reference failure;34=Wrong system
voltage;35=Pre-charge timeout";
$table="0=Off;2=Fault;3=Bulk;4=Absorption;5=Float;6=Storage;7=Equalize;11=Other (Hub-1);252=External control";
$table="0=No error;1=Battery temperature too high;2=Battery voltage too high;3=Battery temperature sensor miswired
(+);4=Battery temperature sensor miswired (-);5=Battery temperature sensor disconnected;6=Battery voltage sense miswired
(+);7=Battery voltage sense miswired (-);8=Battery voltage sense disconnected;9=Battery voltage wire losses too
high;17=Charger temperature too high;18=Charger over-current;19=Charger current polarity reversed;20=Bulk time limit
reached;22=Charger temperature sensor miswired;23=Charger temperature sensor disconnected;34=Input current too high";
$table="0=Unused, BL disabled;1=Restarting;2=Self-consumption;3=Self-consumption;4=Self-consumption;5=Discharge
disabled;6=Force charge;7=Sustain;8=Low Soc Recharge;9=Keep batteries charged;10=BL Disabled;11=BL Disabled (Low
SoC);12=BL Disabled (Low SOC recharge)";
$table="0=OK;1=Disconnected;2=Short circuited;3=Reverse Polarity;4=Unknown";
$table="0=Low;1=High;2=Off;3=On;4=No;5=Yes;6=Open;7=Closed;8=Alarm;9=OK;10=Running;11=Stopped";
$des=explode(";",$table);
foreach($des as $p) {
    $n=explode("=",$p);
    IPS_SetVariableProfileAssociation("Victron.DigitalInputState",$n[0],$n[1],null,-1);
    }
}
```

3. Werte im Cerbo aus IPSymcon verändern

Um eine Variable an den Cerbo zu übertragen muß für jede Variable ein MQTT Client Device erstellt werden.
Das Topic entspricht dabei dem Topic der Variablen, im Topic wird der erste Buchstabe ("N") durch ein "W" ersetzt. Der Typ der Variablen ist immer String.
Hier ein Beispiel:



Das Setzen der Variablen aus einem Script erfolgt dann zum Beispiel mit:

```
RequestAction(43211,'{"value":'$Einspeisung.'}');
```

Wie auf dem Bild ersichtlich ist bei mir das MQTT Device zum Schreiben ein Child von value der Leseinstanz.
Damit kann ich im Webfront die Variable direkt verändern.
Ich habe in Script, das allen Variablen zugweisen wird, die im Webfront veränderbar sein sollen, ein Teil des Script prüft ob der Wert eine Childinstanz hat, und führt dann eine RequestAction statt eine Veränderung der Variablen aus.

```
$p=@IPS_GetInstance(IPS_GetParent($_IPS['VARIABLE']));
if ($p<>null) {
  if ($p["ModuleInfo"]["ModuleID"]=="{91D174F2-AE0F-B8D8-5EF4-6232B9083CCF}") {
    $p1=IPS_GetChildrenIDs($_IPS['VARIABLE']);
    $p2=IPS_GetChildrenIDs($p1[0]);
    RequestAction($p2[0],'{"value":'.$_IPS['VALUE'].'}');
  }
}
```