IBHsoftec

TTi
Trans-Tech International
Ingenieurbüro für Technologie Transfer
Dipl.-Ing. B. Peter Schulz-Heise

IBH OPC UA Editor

MQTT – Konfiguration / TIA

Handbuch

Version 7.4.9

IBHsoftec GmbH
Turmstr. 77
64760 Oberzent / Beerfelden
Tel.: +49 6068 3001
Fax: +49 6068 3074
info@ibhsoftec.com
www.ibhsoftec.com

TTi Ingenieurbüro für
Technologie Transfer
Dipl. Ing. B. Peter Schulz-Heise
Tel.: +49 6061 3382
Fax: +49 6061 71162
tti@ schulz-heise.com
www.schulz-heise.com

Inhalt

MQTT– Konfigurationsbeispielprojekte

SPS-Programme / *IBH OPC UA Editor-Dateien [*.opu]*

MQTT-PLC416-Job 11-03-2022 ***MQTT-PLC416-Job 11.03.2022.opu***	***Beispiel – Projekt: OPC UA S7 Counter*** *STEP® 7 TIA Portal V17; CPU 416-3 PN/DP.*

1 Beispiel: MQTT - Konfiguration

Mit dem folgenden Beispiel wird die Anbindung an Cloud-Dienste via MQTT erläutert.

1.1 Anbindung an Cloud-Dienste:

Mit MQTT ist die Verbindung zu verschiedenen Cloud-Diensten und Anbietern möglich. Dies sind beispielsweise:

- **Microsoft Azure IoT Hub**.

- **IBM BlueMix**.

- **Amazon Webservices (AWS)**.

- Beliebige, lokal oder im Intranet gehostete **MQTT Broker**, wie z.B. **RabbitMQ**.

1.1.1 Beispiel CloudExample – PLC 416

Eine PLC 416 Steuerung empfängt Daten von einem übergeordneten Warenwirtschaftssystem. Das Werkstück wird bearbeitet. Während der Bearbeitung wird kontinuierlich der Bearbeitungsgrad gemeldet.

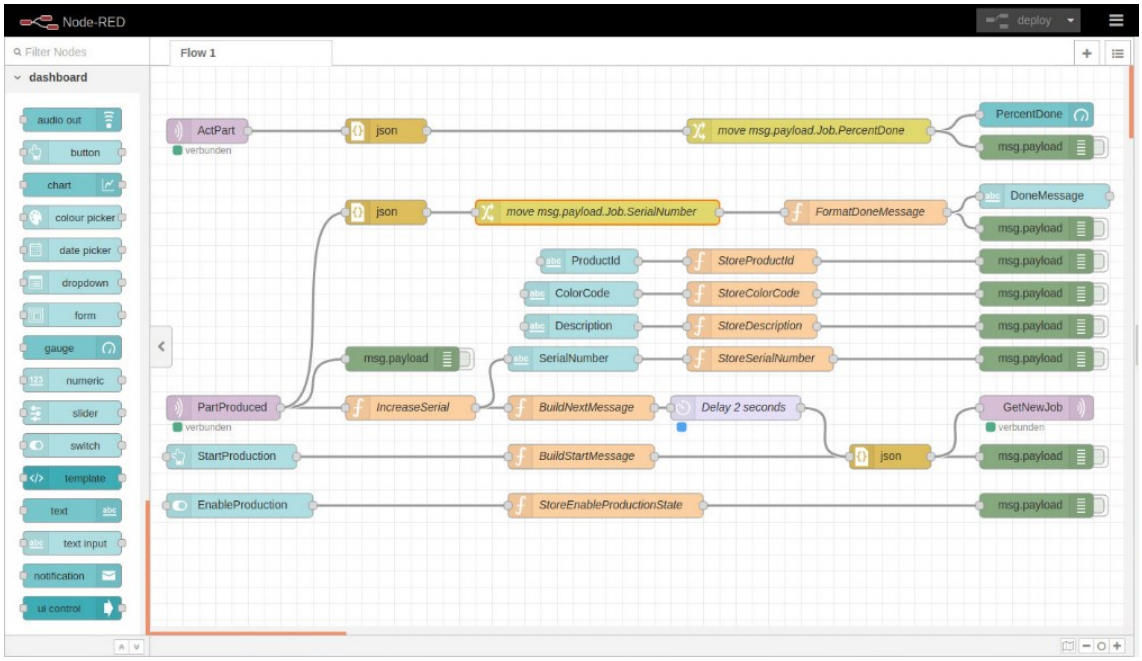
Nach Fertigstellung des Werkstücks wird dies ebenfalls an das übergeordnete System gemeldet. Danach ist die Maschine wieder für einen neuen Auftrag bereit.

Die Demo ist wie folgt aufgebaut:

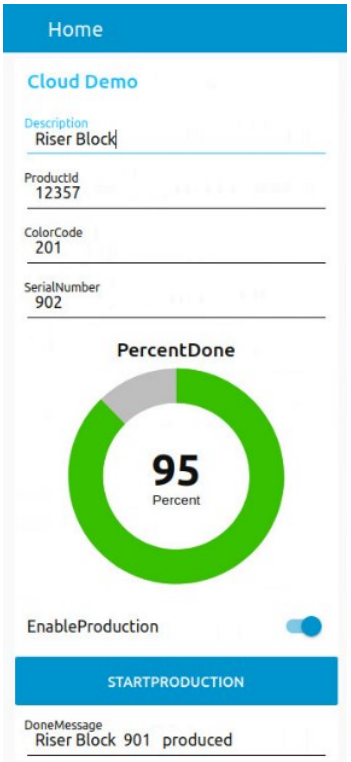
- An der PLC 416 Steuerung ist ein IBH Link UA angeschlossen, der über das MP Protokoll mit der SPS kommuniziert.
- Der IBH Link UA ***publisht Daten*** ein einen **RabbitMQ MQTT Broker**.

- Gleichzeitig hat sich der IBH Link UA für MQTT ***Daten subscribed***.
- Zur anschaulichen Darstellung des Warenwirtschaftssystem wird dieses mit einem übersichtlichen ***Node-RED Flow*** emuliert.

Beispiel Node-RED Flow:



In der Demo wird ***stellvertretend für die Cloud*** ein lokaler ***RabbitMQ*** MQTT Broker betrieben. Dieser ist kostenlos für Linux und Windows unter www.rabbitmq.com erhältlich.



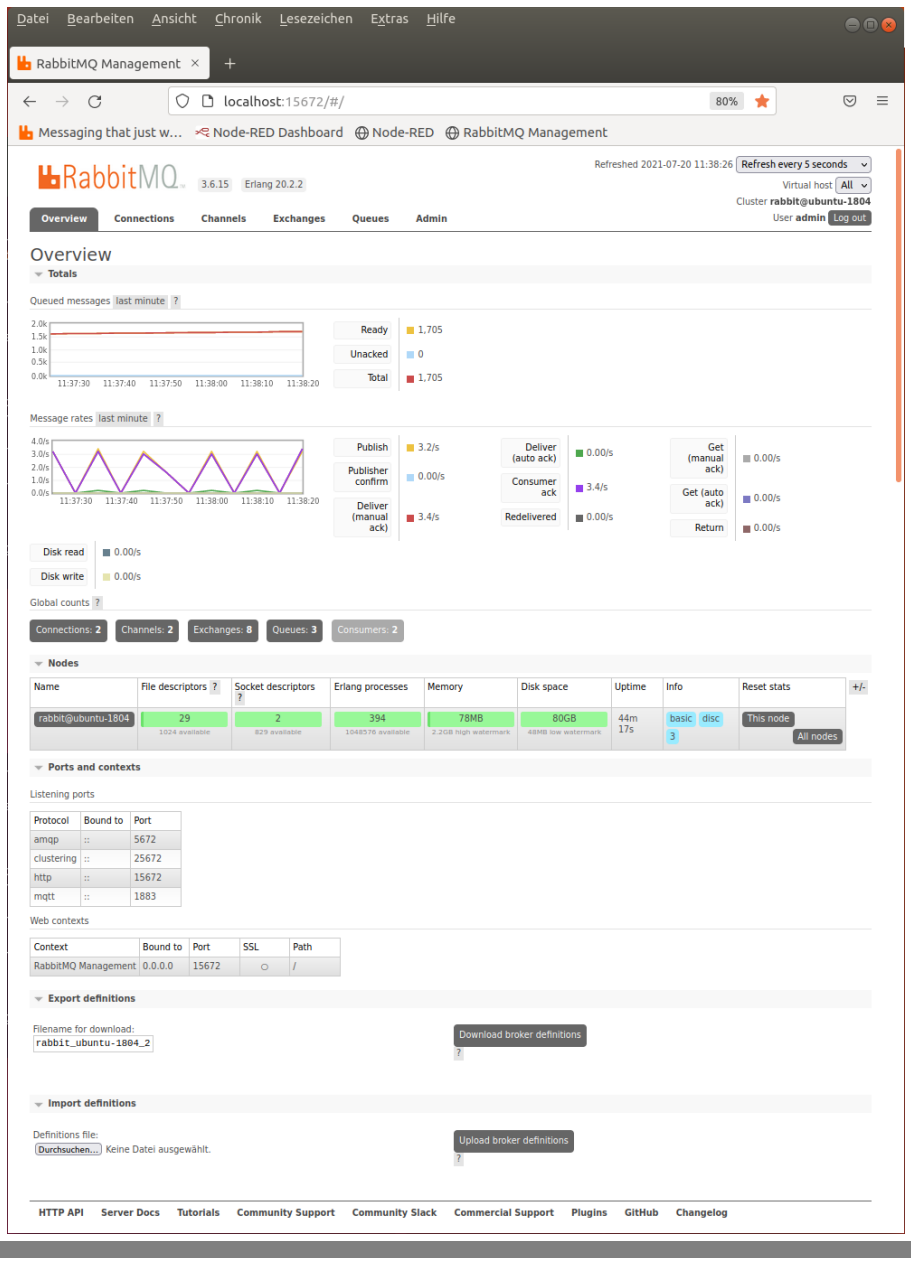
- Das ***MQTT Plugin*** muss nachinstalliert werden, damit der Broker auf dem MQTT Port ***1883*** (default) eingehende Verbindungen von MQTT Clients, wie dem ***IBH Link UA*** und z.B. ***Node-RED*** akzeptiert.
- Einen Benutzer ***ibhsoftec*** mit Passwort ***ibhsoftec*** wurde angelegt, der zu Demozwecken unverschlüsselt kommuniziert.
- Die Anleitungen zum Installieren des RabbitMQ Brokers sowie des MQTT Plugins finden sich ebenfalls unter www.rabbitmq.com.

- Parallel zum RabbitMQ Broker haben wir in der Demo ***Node.js*** und ***Node-RED*** parallel installiert, damit die Flows auf einem leistungsstarken Server ablaufen können. Zudem kann von jedem Rechner mit einem Browser darauf zugegriffen werden.

- Zur Visualisierung haben wir das ***Node-RED*** Plugin ***dashboard*** installiert, um Werte in einer einfachen, webbasierten GUI anzeigen zu können.

- Anleitungen zur Installation für das jeweilige Betriebssystem finden sich im Internet genügend.

- Es ist sinnvoll, in der Steuerung Strukturen anzulegen, die den MQTT Strukturen entsprechen, da dies die Übersichtlichkeit erhöht. Dies ist jedoch nicht zwingend erforderlich.

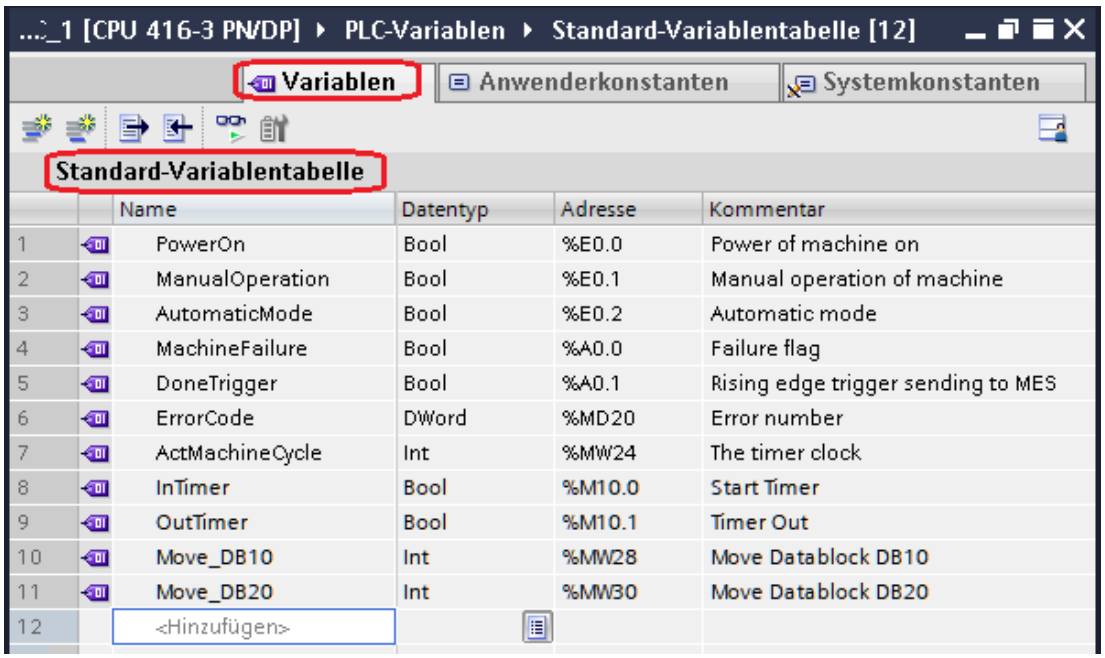


1.2 SPS Programm

Bei der PLC 416 Steuerung sind die Variablen in global adressierbaren Bereichen und in Datenbausteinen (DB 10, DB 20, DB 30) abgelegt.

Standard-Variablentabelle

Hier ist die Variable ***DoneTrigger*** festgelegt.



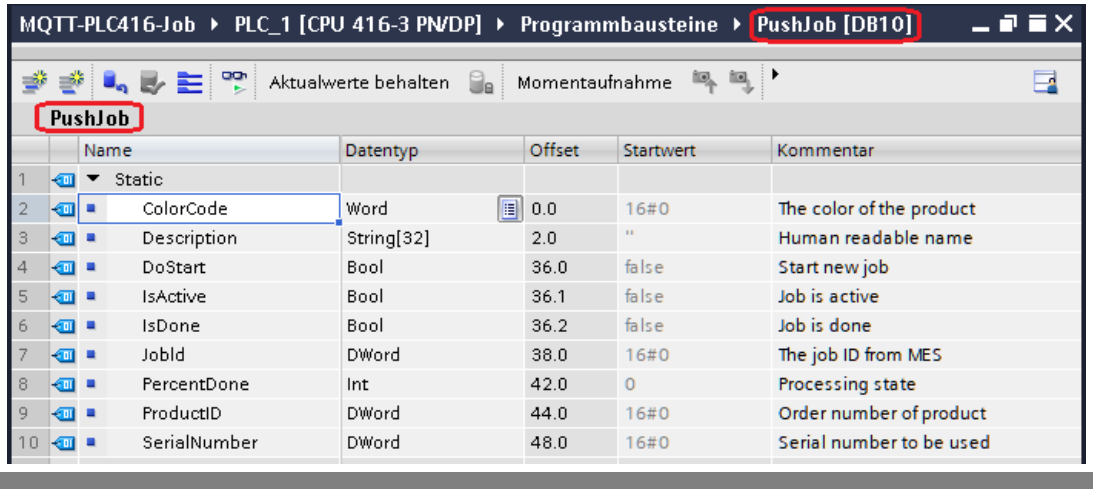
	Name	Datentyp	Adresse	Kommentar
1	PowerOn	Bool	%E0.0	Power of machine on
2	ManualOperation	Bool	%E0.1	Manual operation of machine
3	AutomaticMode	Bool	%E0.2	Automatic mode
4	MachineFailure	Bool	%A0.0	Failure flag
5	DoneTrigger	Bool	%A0.1	Rising edge trigger sending to MES
6	ErrorCode	DWord	%MD20	Error number
7	ActMachineCycle	Int	%MW24	The timer clock
8	InTimer	Bool	%M10.0	Start Timer
9	OutTimer	Bool	%M10.1	Timer Out
10	Move_DB10	Int	%MW28	Move Datablock DB10
11	Move_DB20	Int	%MW30	Move Datablock DB20
12	<Hinzufügen>			

Datenbausteinen DB 10, DB 20, DB 30

Die Datenbausteinen DB 10, DB 20, DB 30 haben alle die gleichen ***Datenstrukturen***.

Datenbausteine *PushJob* (DB 10) - *Daten subscribed*

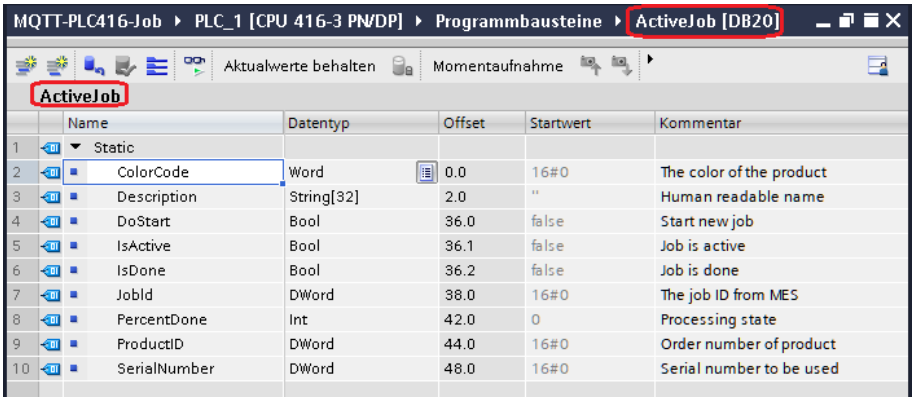
Der Datenbaustein DB 10 (PushJob) bekommt die Werte der Variablen von dem IBH Link UA aufgrund der ***Daten subscribed*** für MQTT.



	Name	Datentyp	Offset	Startwert	Kommentar
1	Static				
2	ColorCode	Word	0.0	16#0	The color of the product
3	Description	String[32]	2.0	''	Human readable name
4	DoStart	Bool	36.0	false	Start new job
5	IsActive	Bool	36.1	false	Job is active
6	IsDone	Bool	36.2	false	Job is done
7	JobId	DWord	38.0	16#0	The job ID from MES
8	PercentDone	Int	42.0	0	Processing state
9	ProductID	DWord	44.0	16#0	Order number of product
10	SerialNumber	DWord	48.0	16#0	Serial number to be used

Datenbausteinen ActiveJob (DB 20) – *Daten published*

Das SPS-Programm der PLC416 überträgt die Daten des Datenbausteins *PushJob* (DB 10) aufgrund der subscribed Variablen ***"PushJob".DoStart***.



	Name	Datentyp	Offset	Startwert	Kommentar
1	Static				
2	ColorCode	Word	0.0	16#0	The color of the product
3	Description	String[32]	2.0	''	Human readable name
4	DoStart	Bool	36.0	false	Start new job
5	IsActive	Bool	36.1	false	Job is active
6	IsDone	Bool	36.2	false	Job is done
7	JobId	DWord	38.0	16#0	The job ID from MES
8	PercentDone	Int	42.0	0	Processing state
9	ProductID	DWord	44.0	16#0	Order number of product
10	SerialNumber	DWord	48.0	16#0	Serial number to be used

Datenbausteinen JobDone (DB 30) – *Daten published*

Das SPS-Programm der PLC416 überträgt die Daten des Datenbausteins *ActiveJob* (DB 20) aufgrund der Variablen ***"ActiveJob".IsActive***.

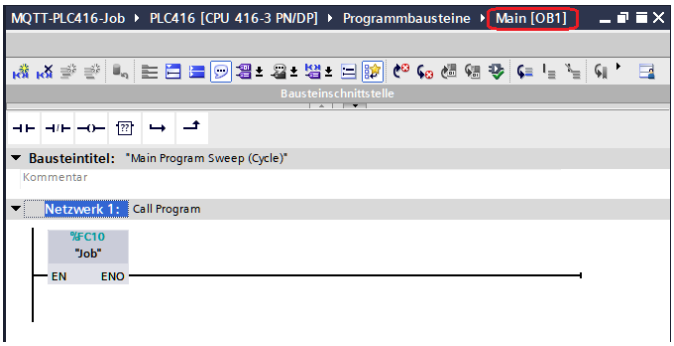
	Name	Datentyp	Offset	Startwert	Kommentar
1	Static				
2	ColorCode	Word	0.0	16#0	The color of the product
3	Description	String[32]	2.0	''	Human readable name
4	DoStart	Bool	36.0	false	Start new job
5	IsActive	Bool	36.1	false	Job is active
6	IsDone	Bool	36.2	false	Job is done
7	JobId	DWord	38.0	16#0	The job ID from MES
8	PercentDone	Int	42.0	0	Processing state
9	ProductID	DWord	44.0	16#0	Order number of product
10	SerialNumber	DWord	48.0	16#0	Serial number to be used

1.2.1 SPS-Programm MQTT-PLC416-Job

In der Steuerung ist zu Demozwecken ein vereinfachtes Programm.

Main [OB 1]

Hier wird die Funktion **Job (FC 10)** aufgerufen. Weitere Befehle sind für das einfache Demo-Programm nicht notwendig.



Funktion FC10 (Job)

MQTT-PLC416-Job AWL ▸ PLC_1 [CPU 416-3 PN/DP] ▸ Programmbausteine ▸ Job [FC10]

```
1  // Simulate Power on
2  IF ("PowerOn" = FALSE) THEN
3      "PowerOn" := TRUE;
4  END_IF;
5  // Simulate Automatic mode
6  IF ("AutomaticMode" = FALSE) THEN
7      "AutomaticMode" := TRUE;
8  END_IF;
9  // Simulate no errors
10 "ManualOperation" := FALSE;
11 "MachineFailure" := FALSE;
12 "ErrorCode" := 0;
13
14 "OnDelay".TOF(IN := "InTimer",
15                PT := T#600ms,
16                Q=>"OutTimer");
17
18 IF ("OnDelay".STATE = 16#01) THEN
19     "InTimer" := False;
20 END_IF;
21     IF ("OnDelay".STATE = 16#00) THEN
22         "ActMachineCycle" := "ActMachineCycle" + 1;
23         "InTimer" := TRUE;
24 END_IF;
25
26  // New Job from MES
27 IF ("PushJob".DoStart) THEN
28     // Reset previos Job
29     "DoneTrigger" := FALSE;
30     "JobDone".ProductID := 0;
31     "JobDone".SerialNumber := 0;
32     "JobDone".ColorCode := 0;
33     "JobDone".JobId := 0;
34     "JobDone".PercentDone := 0;
35     "JobDone".DoStart := FALSE;
36     "JobDone".IsActive := FALSE;
37     "JobDone".IsDone := 0;
38     "JobDone".Description := ' ';
39     // Activate new Job Transfer Data to ActivJob
40     "PushJob".PercentDone := 0;
41     "Move_DB10" := BLKMOV
42     (SRCBLK := "PushJob",
43      DSTBLK => "ActiveJob");
44     "PushJob".DoStart := FALSE;
45     "ActMachineCycle" := 0;
46     "ActiveJob".IsActive := TRUE;
47 END_IF;
48     // Job is being processed;
49 IF ("ActiveJob".IsActive) THEN
50         IF "ActMachineCycle" <= 20 THEN
51             "ActiveJob".PercentDone := "ActMachineCycle" * 5;
52         ELSE
53             "ActiveJob".IsActive := FALSE;
54             "ActiveJob".IsDone := TRUE;
55             "Move_DB20" := BLKMOV
56             (SRCBLK := "ActiveJob",
57              DSTBLK => "JobDone");
58             "ActiveJob".PercentDone := 0;
59         END_IF;
60     END_IF;
61
62     IF ("JobDone".IsDone = TRUE) THEN
63         "DoneTrigger" := TRUE;
64     END_IF;
65
```

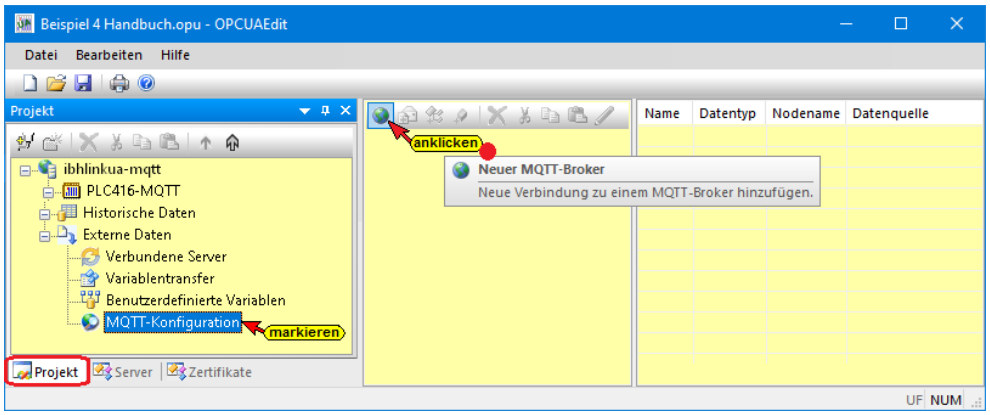
- Es wird immer ***PowerOn*** gemeldet.
- Es wird immer ***AutomaticMode*** ein gemeldet.
- Die Variable ***CountUP*** zählt ***ActMachineCycle*** hoch. Diese wird später zur Simulation eines Produktionsvorganges verwendet.

- ***PushJob.DoStart*** setzt ***DoneTrigger*** zurück und löscht den vorherigen Auftrag. Danach wird der neue Auftrag in ***ActiveJob*** verschoben und die Produktion wird gestartet.
- Während ***ActiveJob*** läuft, wird aus dem Zähler ***ActMachineCycle*** eine Prozentanzeige gebildet, um die Produktion zu simulieren.
- Sind 100% erreicht wird ***ActiveJob*** nach ***JobDone*** verschoben.
- Danach wird D***oneTrigger*** gesetzt, um ein ***MQTT Publish*** auszulösen.

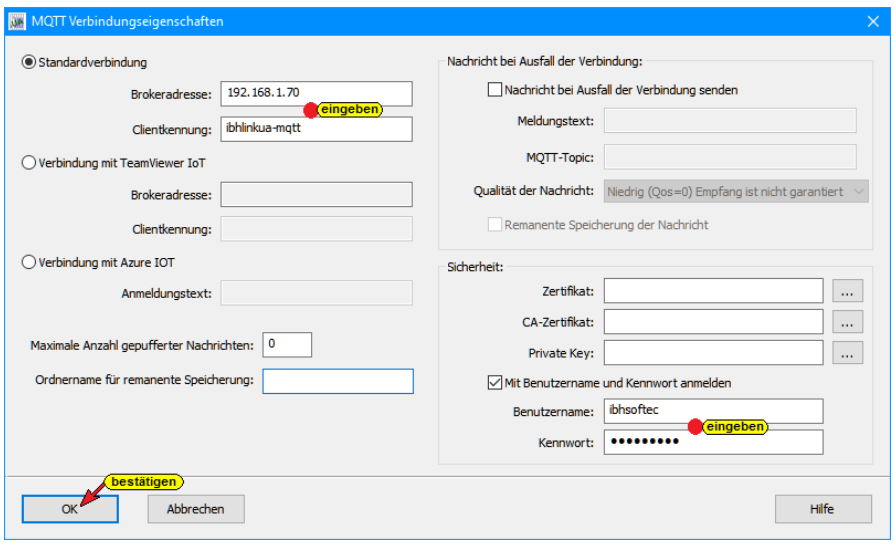
In dem Programm sind zu Demozwecken keine Sicherheitsabfragen realisiert.

Verbindung zum MQTT Broker einfügen

Im linken Teil des ***Projekt-Fensters*** das Symbol ***MQTT Konfiguration*** markieren.



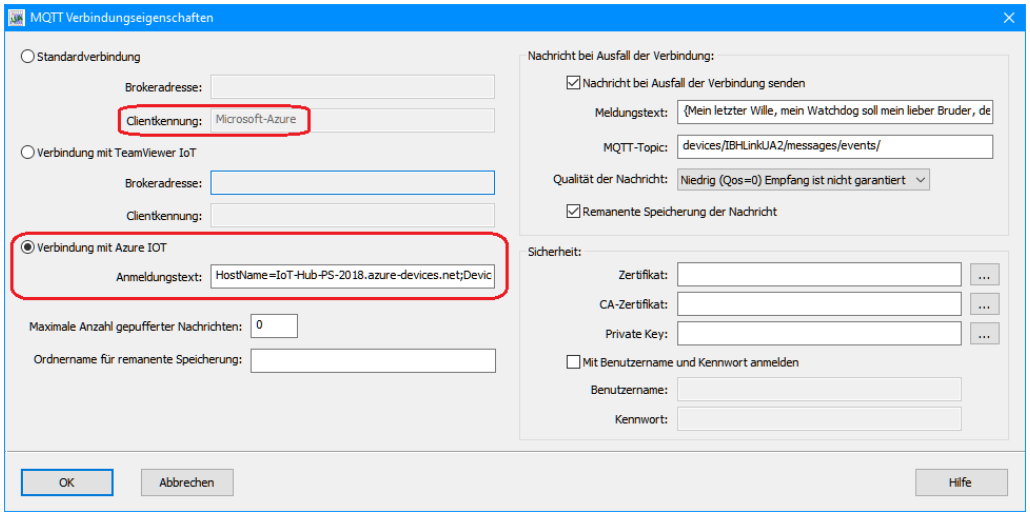
Mit Anklicken des Symbols ***Neuer MQTT-Broker*** wird das Dialogfeld zur Konfiguration eines ***MQTT-Brokers*** geöffnet.



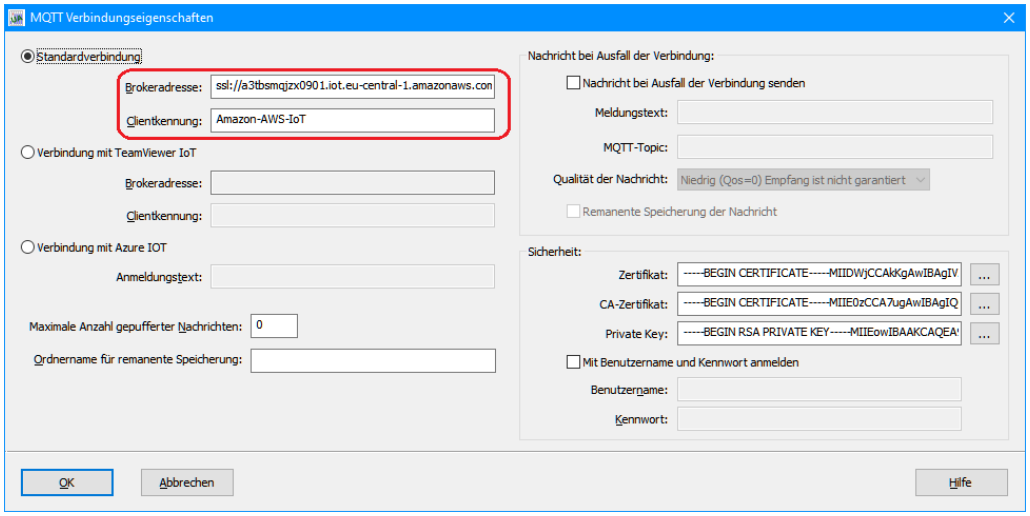
Die Rückfrage, ob bestehende Daten übernommen werden sollen, kann bei der Erstkonfiguration verneint werden.
Um das Beispiel einfach zu halten, wird hier nur eine unverschlüsselte Standardverbindung eingerichtet.
Verschlüsselte Verbindungen, sowie Verbindungen zu Azure IOT sind natürlich möglich.

Beispiel: Verbindung zum Microsoft Azure IoT

Zur Verbindung mit dem Azure IoT benötigt man nur den ***Anmeldetext*** (AzureIoTConnectionString). Der Rest ist in der Software verankert. Pro Azure IoT Hub ist nur ein Topic möglich.



Beispiel: Amazon AWS IOT



Damit bei Verbindungsabbruch keine Messages verloren gehen, besteht die Möglichkeit einer remanenten Speicherung.

Ist eine Micro-SD Karte eingesetzt, werden Messages remanent auf der eingesetzten Karte gespeichert.

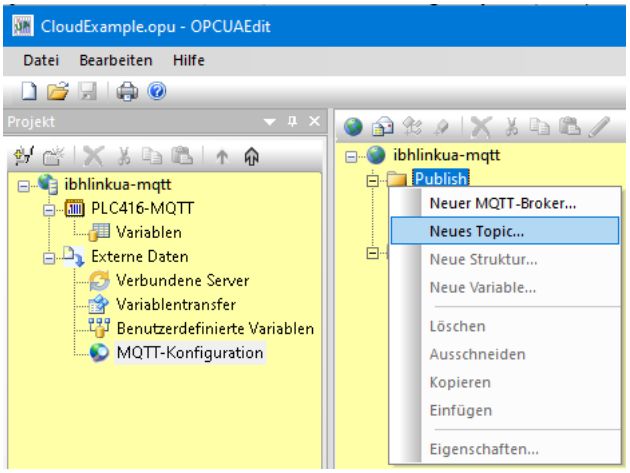
Ebenso kann eine Nachricht bei Verbindungsabbruch definiert werden (Last Will Message).

Topics anlegen

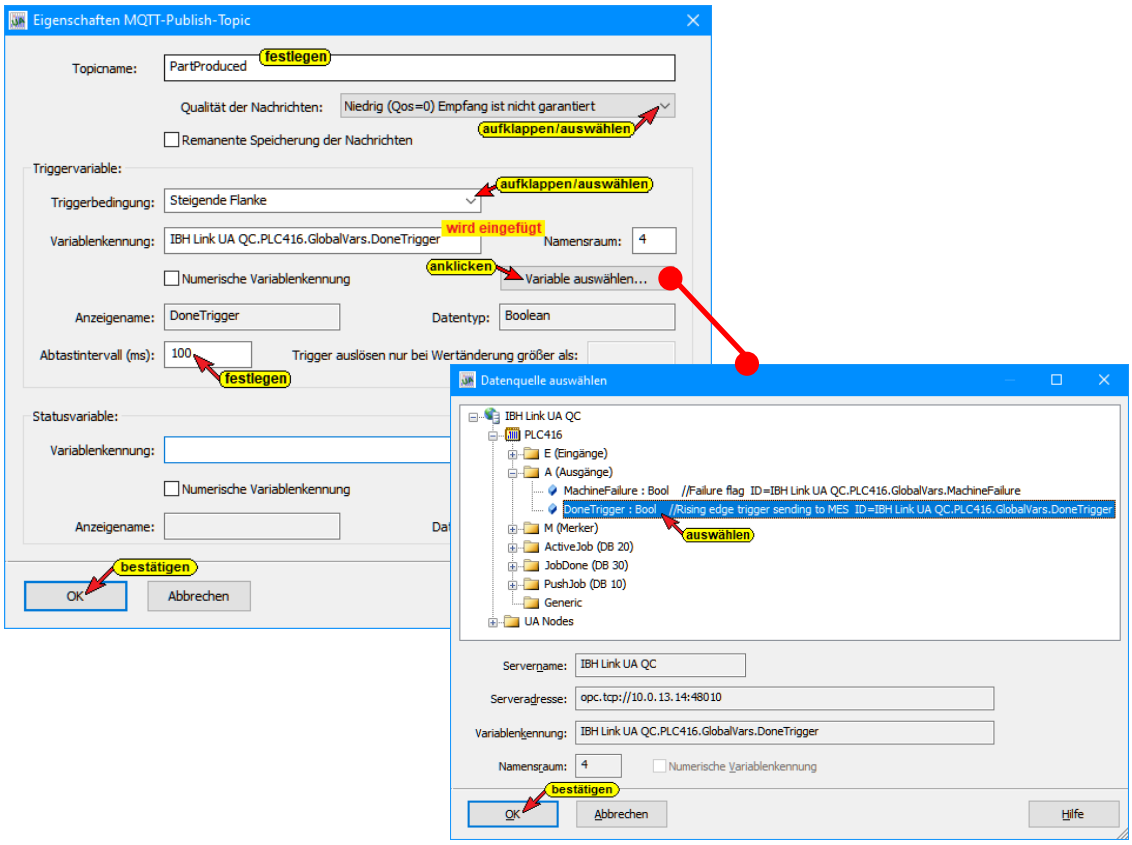
MQTT (Message Queuing Telemetry Transport) verwendet Topics (eindeutige Namen) unter denen die Struktur der zu übermittelnden Daten definiert ist.

- **Publish:** Daten, die zum Broker gesendet werden sollen. Welches Ereignis das Publish auslöst, wird hier festgelegt.
- **Subscribe:** Daten, für die man sich beim Broker interessiert. Bei Änderung übermittelt der Broker die neuen Daten.

Das Prinzip ähnelt Facebook: Jemand postet etwas und mehrere können sich dafür interessieren.

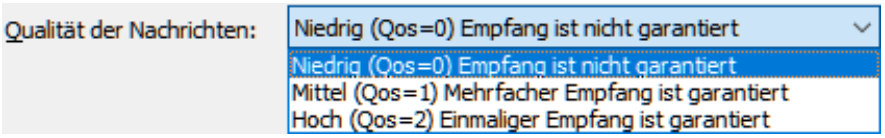


Publish Topic definieren



Topicname: Eindeutiger Name auf dem MQTT Broker.

Der Qos Level (Quality of service) sollte passend zum Broker eingestellt werden, da nicht jeder Broker jeden Level unterstützt.



Triggervariable: Auf Änderung einer Variablen in der Steuerung wird ein Publish ausgelöst. Dies kann für einen 1s Takt auch die Serverzeit sein.

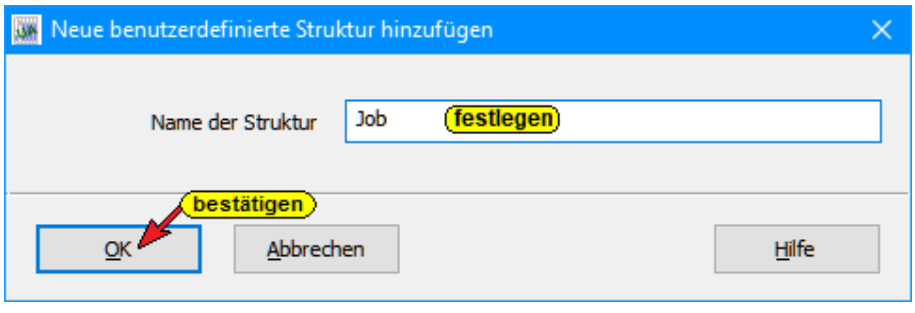
Statusvariable: Der MQTT Status kann in eine SPS Variable gelegt werden, damit das SPS Programm auswerten kann, ob die Message versendet wurde.

Publish Topic modellieren

Da in aller Regel der Aufbau des Topics bereits durch die Applikation und die verwendeten Modelle vorgegeben ist, kann der Aufbau des Topics frei definiert werden.
So kann das Topic an die Anforderungen angepasst werden.
Es können innerhalb des Topics Strukturen, Unterstrukturen und Variablen beliebig geschachtelt werden.

Anlegen einer Struktur bzw. Unterstruktur.

Eine Variable wird mit Namen und Datentyp, sowie einer Beschreibung und Zugriffsrechten definiert.



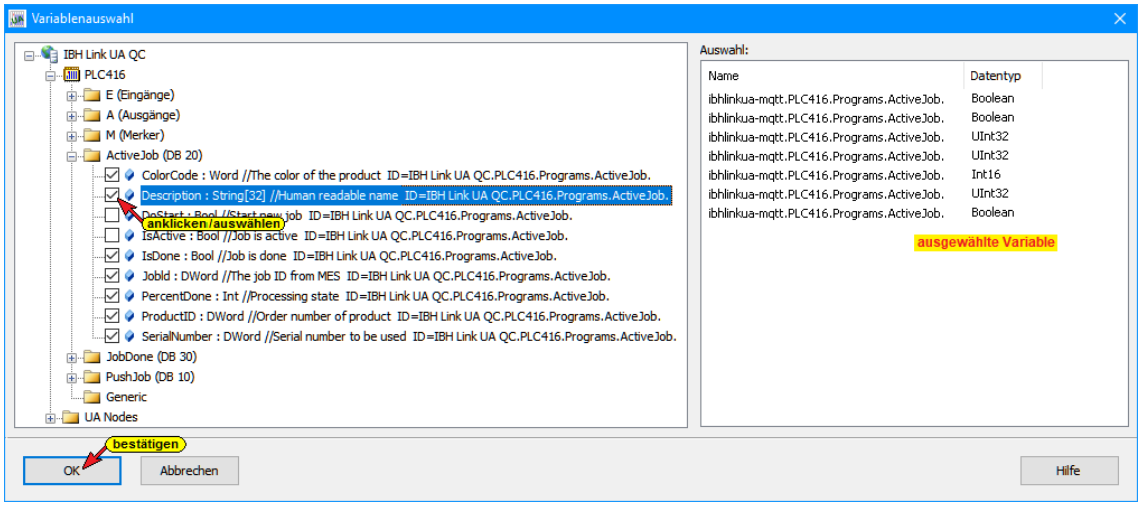
Eine Variable wird mit Namen und Datentyp, sowie einer Beschreibung und Zugriffsrechten definiert.

Variable auswählen

Um eine Variable in eine Struktur bzw. Topic festzulegen, kann dies mit dem Befehle ***Variable auswählen*** aus dem Kontextmenü des

Topics heraus aufgerufen werden. Der Befehl ***Variable auswählen*** öffnet die Auswahlbox ***Variablenauswahl***.
Hier können die gewünschte Variable aus den Variablen der vorhandenen Steuerungen ausgewählt werden. Die ausgewählten Variablen werden direkt in das vorhandene Topic bzw. in die Struktur eingetragen.

Auswahlbox Variablenauswahl

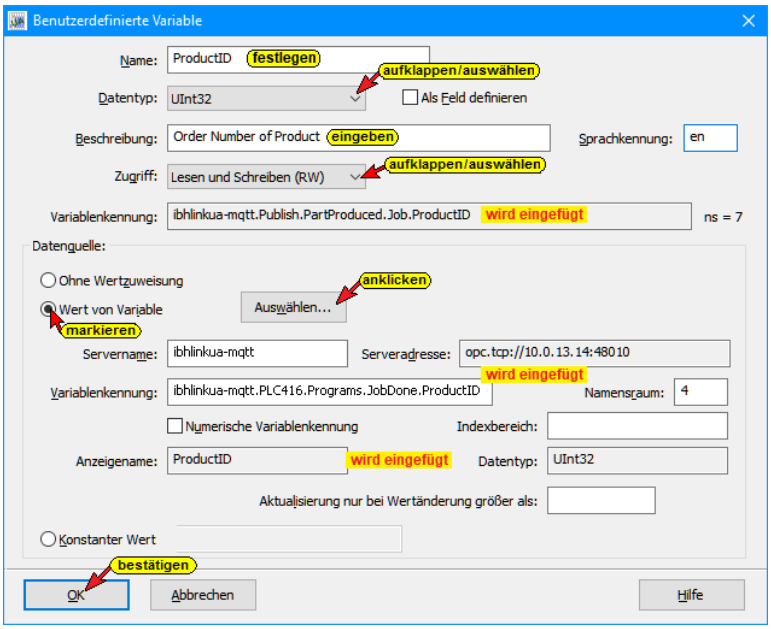


Dieses Auswahlverfahren ist günstig, wenn mehrere Variable einem Topic bzw. einer Struktur zugewiesen werden sollen.

Neue Variable

Um eine Variable in eine Struktur bzw. Topic festzulegen, kann dies mit dem Befehle ***Neue Variable*** aus dem Kontextmenü des Topics heraus aufgerufen werden.

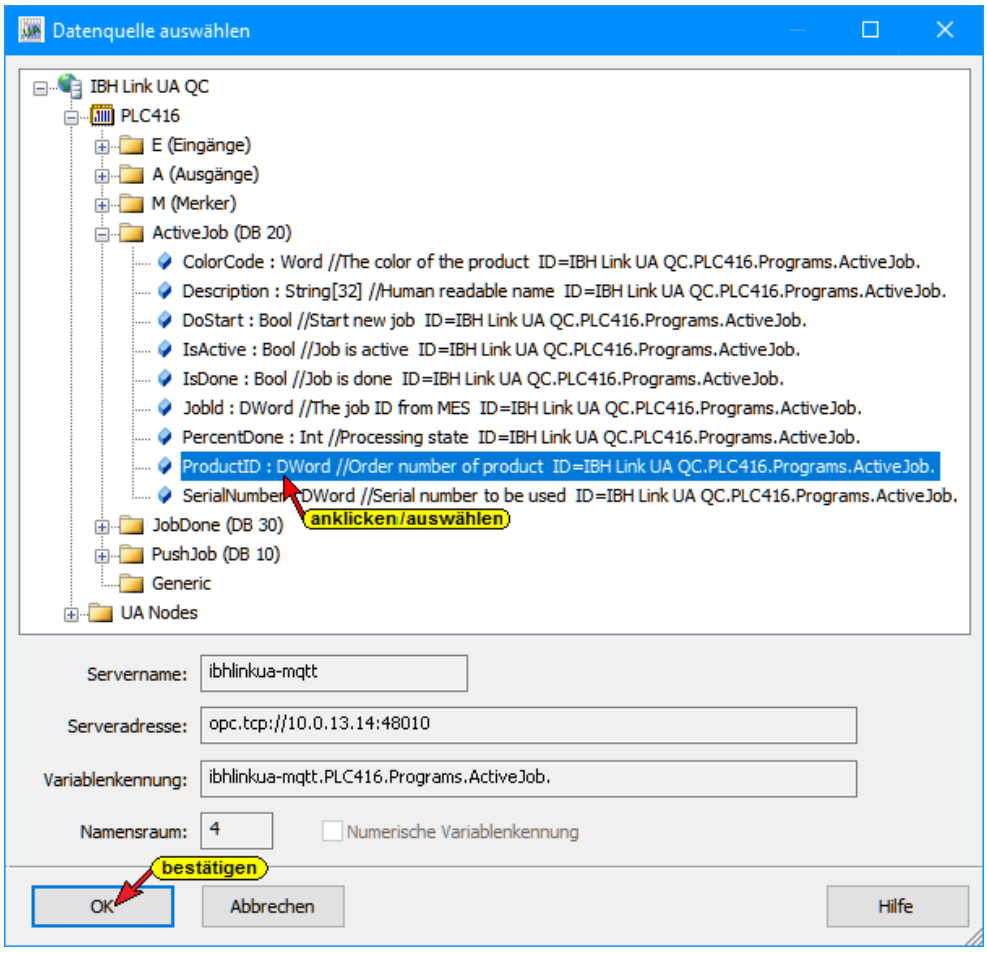
Der Befehl ***Neue Variable*** öffnet die Dialogbox ***Benutzerdefinierte Variable***. Hier kann eine neue Variable definiert werden.



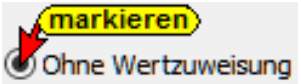
Die Schaltfläche Auswählen öffnet die Auswahlbox Datenquelle auswählen. Hier kann eine Variable aus den Variablen der vorhandenen Steuerungen ausgewählt werden.

Der Wert der Variablen hat:

- Keine Zuweisung (Wird nicht verändert).
- Eine Zuordnung mit einer Variablen aus der Steuerung oder des OPC Servers. Wird also mit einem Wert der SPS „verdrahtet“.
- Einen konstanten Wert.
- Diese wird seinen Wert an die benutzerdefinierte Variable übergeben.



Eine Variable ohne Datenquelle kann auch festgelegt werden. In diesem Fall muss ***Ohne Wertzuweisung*** markiert werden.



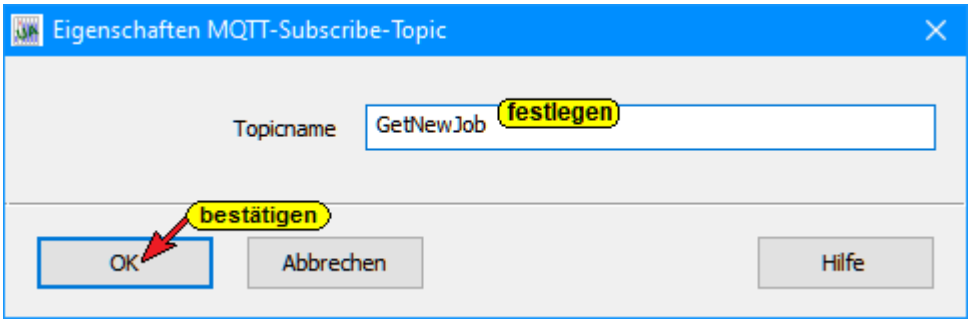
Subscribe Topic definieren

Eine Besonderheit beim IBH Link UA ist, dass dieser nicht nur Werte in die Cloud publishen kann, sondern auch die Möglichkeit bietet, sich für bestimmte Werte beim Broker zu interessieren, sich zu Subscriben.

Diese Werte können dann unmittelbar in die Steuerung weitergeleitet werden.

Somit lassen sich auftragsbasierte Schnittstellen zur Maschine schaffen.

Topicname: Eindeutiger Name auf dem MQTT Broker. Im Gegensatz zum Publish Topic gibt es hier keine Trigger Bedingung, da das Topic bei Änderung vom MQTT Broker an den IBH Link UA übertragen wird.

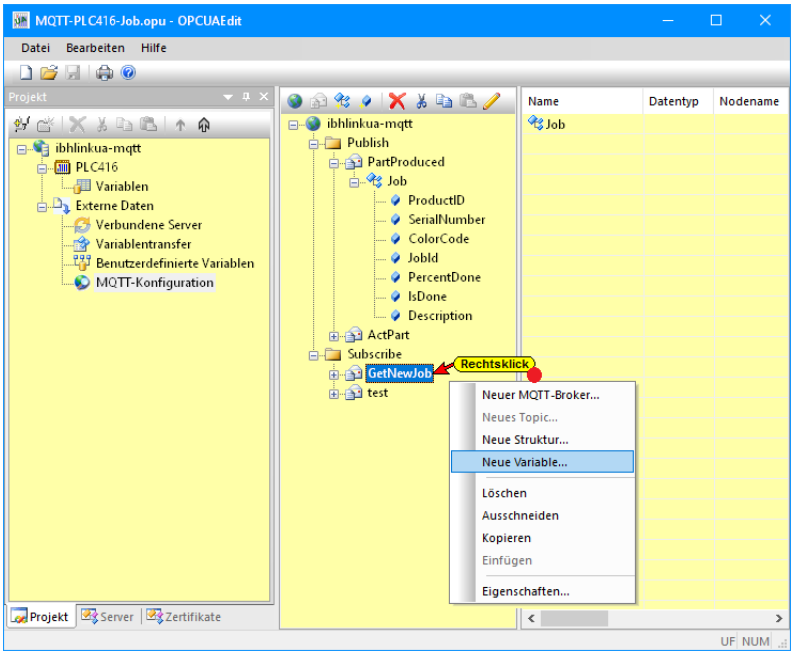


Wie beim Publish Topic auch, ist in aller Regel der Aufbau des Topics bereits durch die Applikation und die verwendeten Modelle vorgegeben. Daher kann auch hier der Aufbau des Topics frei definiert werden.

Es können, wie beim Publish Topic auch, innerhalb des Topics Strukturen, Unterstrukturen und Variablen beliebig geschachtelt werden.

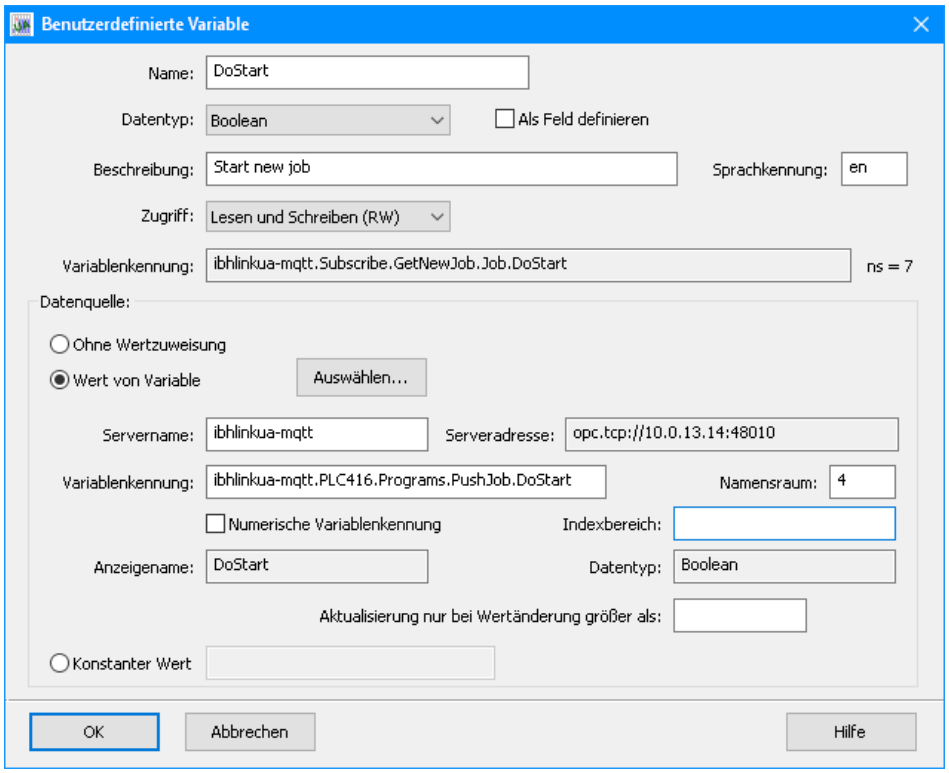
Neue Variable definieren

Eine Variable wird mit Namen und Datentyp, sowie einer Beschreibung und Zugriffsrechten definiert.



Der Wert der Variablen hat:

- Keine Zuweisung (Wird nicht verändert).
- Eine Zuordnung mit einer Variablen aus der Steuerung oder des OPC Servers. Wird also mit einem Wert der SPS „verdrahtet“.
- Einen konstanten Wert.



Anmerkung zu Topics:

Es ist keine Datenkonsistenz gewährleistet !

Die SPS Anwendung muss für die Datenkonsistenz sorgen.

Ein ***Publish Topic*** ohne eine ***Triggerbedingung*** legt auf dem OPC UA Server eine ***Subscription*** an, die vom Server im ***Abtastintervall (ms)*** des Topics aktualisiert wird. Dies bedeutet, die zu publizierenden Werte kommen aus einem Cache.

Wird eine ***Triggervariable*** definiert, wird nur diese im ***Abtastintervall (ms)*** des Topics aktualisiert. Bei Auslösen des Triggers werden alle Variablen des Topics zwingend gelesen.

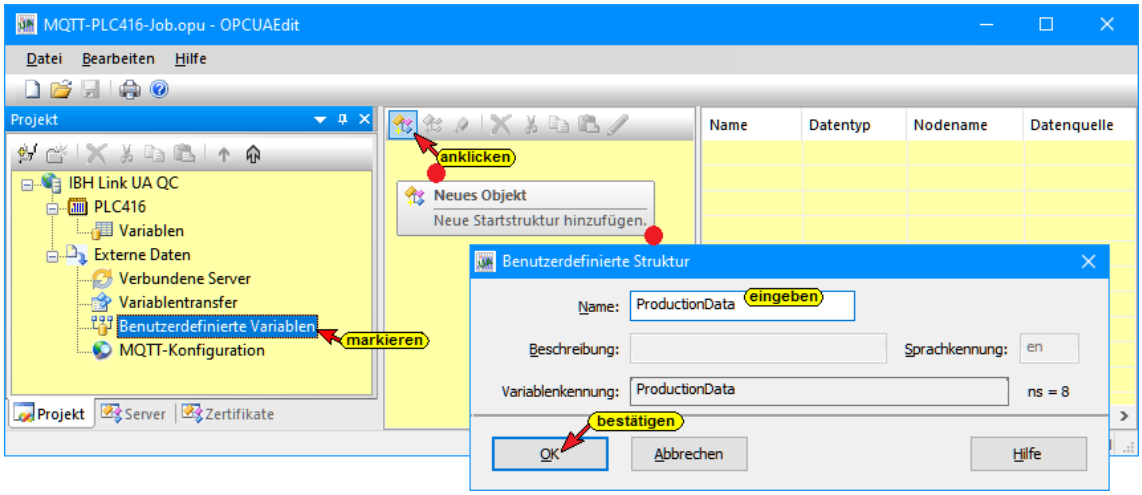
Eine ***Subscribe Topic*** schreibt die Werte so optimiert als möglich in die Steuerung. Da nicht zwingend alle Daten in eine Message zu Steuerung passen, kann dies in mehreren Frames geschehen. Abhängig von der Steuerung und der internen Optimierung kann daher nicht vorhergesagt werden, welcher Wert zuerst ankommt.

Die SPS Anwendung muss für die Datenkonsistenz sorgen.

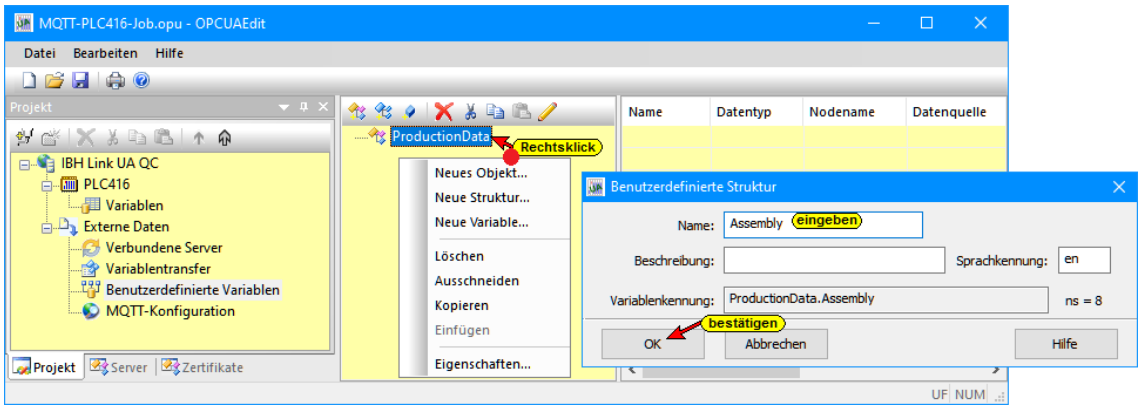
Tip: 2x Schreiben (einmal alle Daten ***ohne Start Flag***, dann alle Daten ***mit Start Flag***) kann funktionieren, da immer ein ***Schreibrequest*** komplett verarbeitet wird.

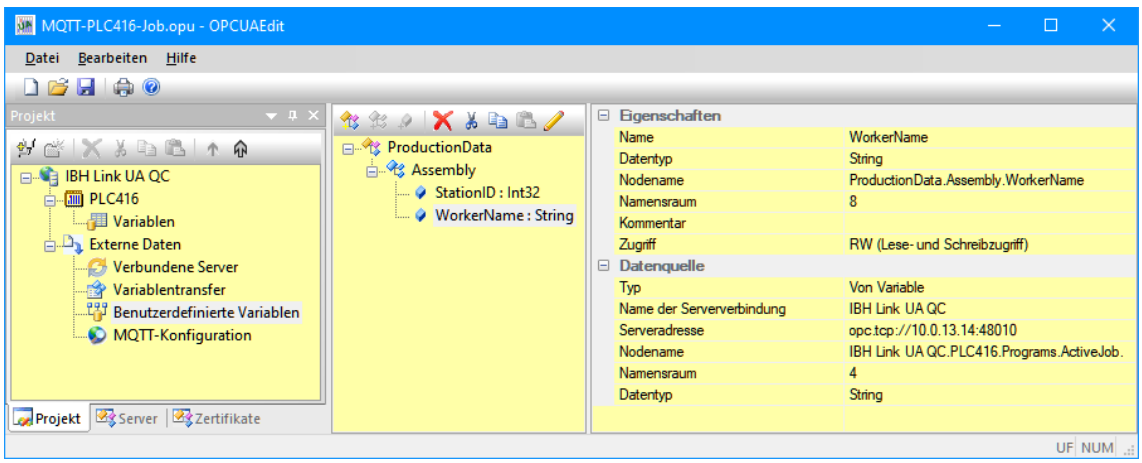
Benutzerdefinierte OPC UA Variablen definieren

Wie beim Erzeugen eines Topics für MQTT, besteht ebenfalls die Möglichkeit die daten für OPC UA benutzerdefiniert zu organisieren und mit SPS Variablen zu verdrahten.



Die Vorgehensweise ist hierbei identisch mit der Vorgehensweise bei MQTT.

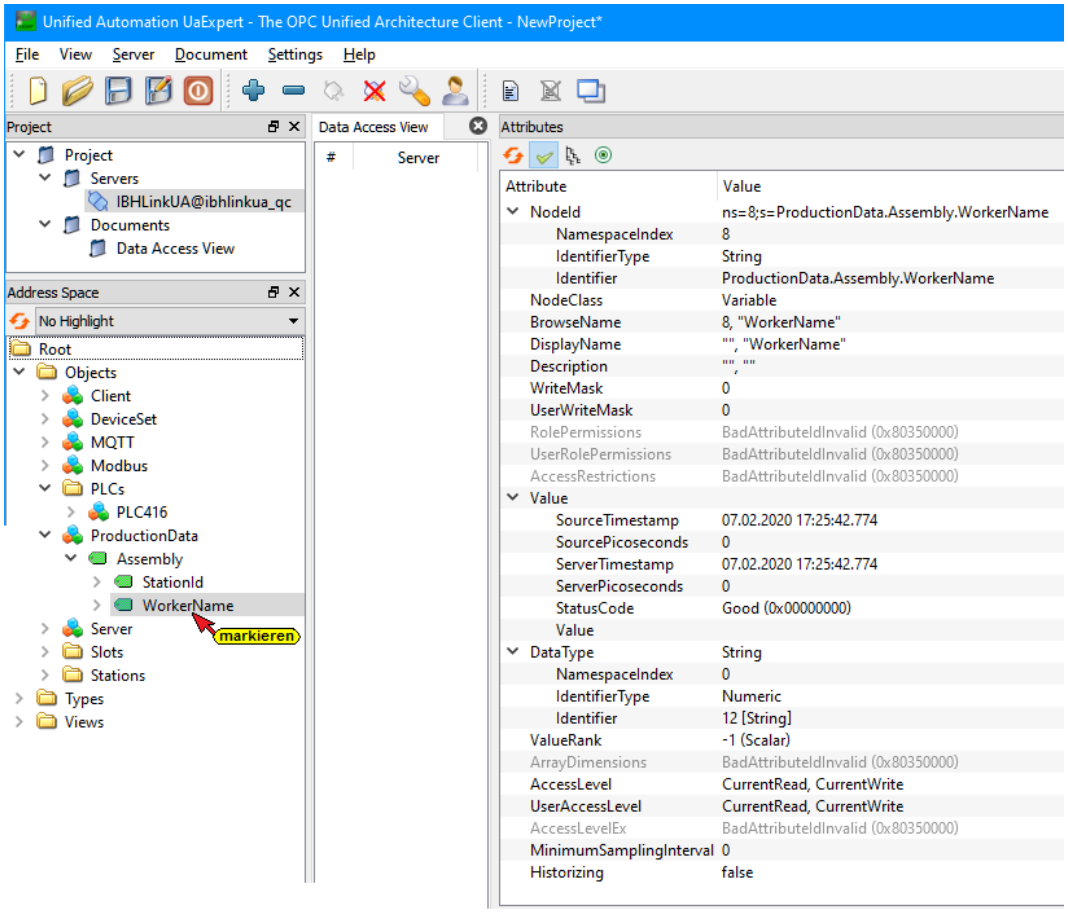




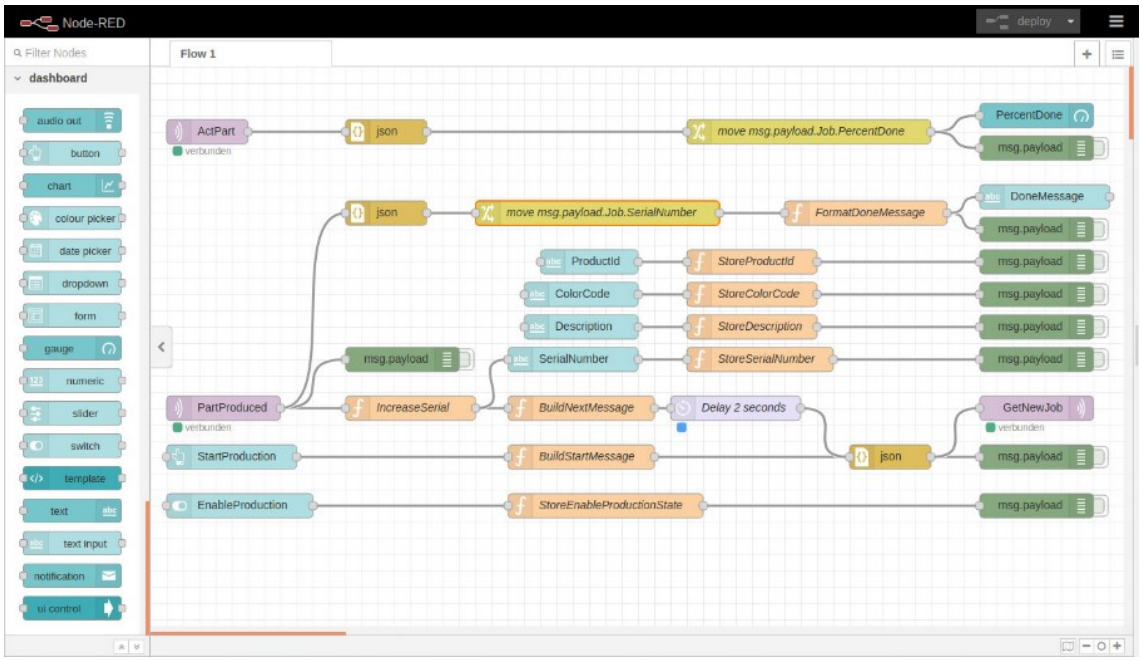
Die benutzerdefinierten Variablen erscheinen im OPC UA Namespace 8 direkt im Namensraum des Servers.

OPC Clients können diese Modellierung verwenden, um kundenspezifische, standardisierte Daten in OPC UA umzusetzen.

UaExpert - Attributes

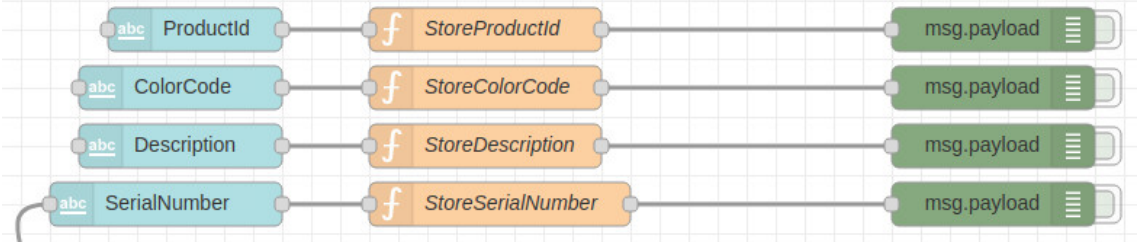


Beschreibung des Beispiels – *Node-RED*

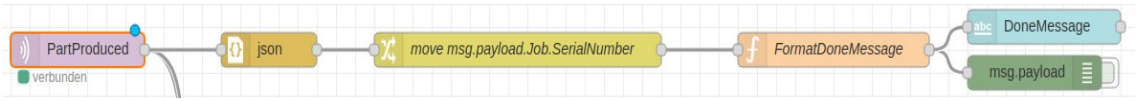


- Der ***mqtt-in*** Node ist auf dem RabbitMQ MQTT Broker für das Topic ***ActPart*** subscribed, um den Fortschritt des Produktionsprozesses auf einer Skala anzuzeigen.
- Der ***json*** Node wandelt den ankommenden json (Java Script Object Notation) String in ein ***json*** Objekt, damit mit ***Objektname.Struktur.Element*** auf die Elemente zugegriffen werden kann.
- Der ***change*** Node extrahiert aus ***payload.Job.PercentDone*** den aktuellen Prozentwert.
- Der ***gauge*** Node zeigt in der Weboberfläche den Prozentsatz an.
- Der ***debug*** Node dient nur der Inbetriebnahme.

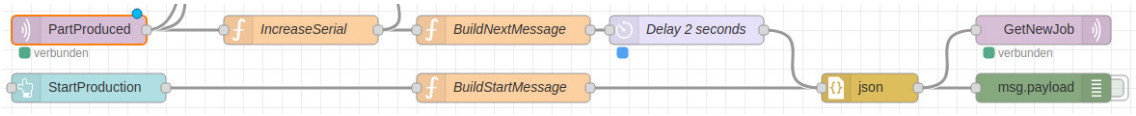
- Der ***switch*** Node ist ein Schalter in der Weboberfläche für einen kontinuierlichen Produktionsprozess. Dieser Simuliert den Automatikbetrieb eines übergeordneten Systems., damit mit ***Objektname.Struktur.Element*** auf die Elemente zugegriffen werden kann.
- Der ***function*** Nodes ***StoreEnableProductionState*** enthält Code, um die Stellung des Schalters in – einer für diesen Flow lokalen - Variablen zu speichern.
- Der ***debug*** Node dient nur der Inbetriebnahme.



- Die ***text input*** Nodes ***ProductId***, ***ColorCode***, ***Description*** und ***SerialNumber*** dienen in der Weboberfläche zur Eingabe der Produktionsdaten. Dies simuliert den Auftrag, der von einem Warenwirtschaftssystem generiert werden würde.
- Die ***function*** Nodes ***Store…*** enthalten Code, um die eingegebenen Daten in - für diesen Flow lokalen - Variablen zu speichern.
- Die ***debug*** Nodes dienen nur der Inbetriebnahme.



- Der ***json*** Node wandelt den ankommenden json (Java Script Object Notation) String in ein ***json*** Objekt, damit mit ***Objektname.Struktur.Element*** auf die Elemente zugegriffen werden kann.
- Der ***change*** Node extrahiert aus ***payload.Job.PercentDone*** die aktuelle Seriennummer.
- Der ***function*** Nodes ***FormatDoneMessage*** enthält Code, um die Nachricht menschenlesbar auszugeben.
- Der ***text input*** Node zeigt in der Weboberfläche das produzierte Teil an.
- Der ***debug*** Node dient nur der Inbetriebnahme.
- Der ***mqtt-in*** Node ist auf dem RabbitMQ MQTT Broker für das Topic ***PartProduced*** subscribed, um die Fertigstellung des Produktionsprozesses anzuzeigen.



- Der mqtt-in Node ist für das Topic PartProduced subscribed.
- Der function Nodes IncreaseSerial erhöht die Seriennummer um 1.
- Der button Node StartProduction startet die Produktion.
- Die function Nodes BuildStartMessage und BuildNextMessage bilden aus den eingegebenen Informationen die MQTT Message für den nächsten Auftrag (Stellvertretend für das darüberliegende MES System).
- Der delay Node verzögert lediglich zur leichteren Veranschaulichung des laufenden Prozesses.
- Der json Node wandelt das ankommende json (Java Script Object Notation) Objekt in einen json String, damit dieser mit dem mqtt-out Node, an den MQTT Broker versendet werden kann.
- Der debug Node dient nur der Inbetriebnahme.

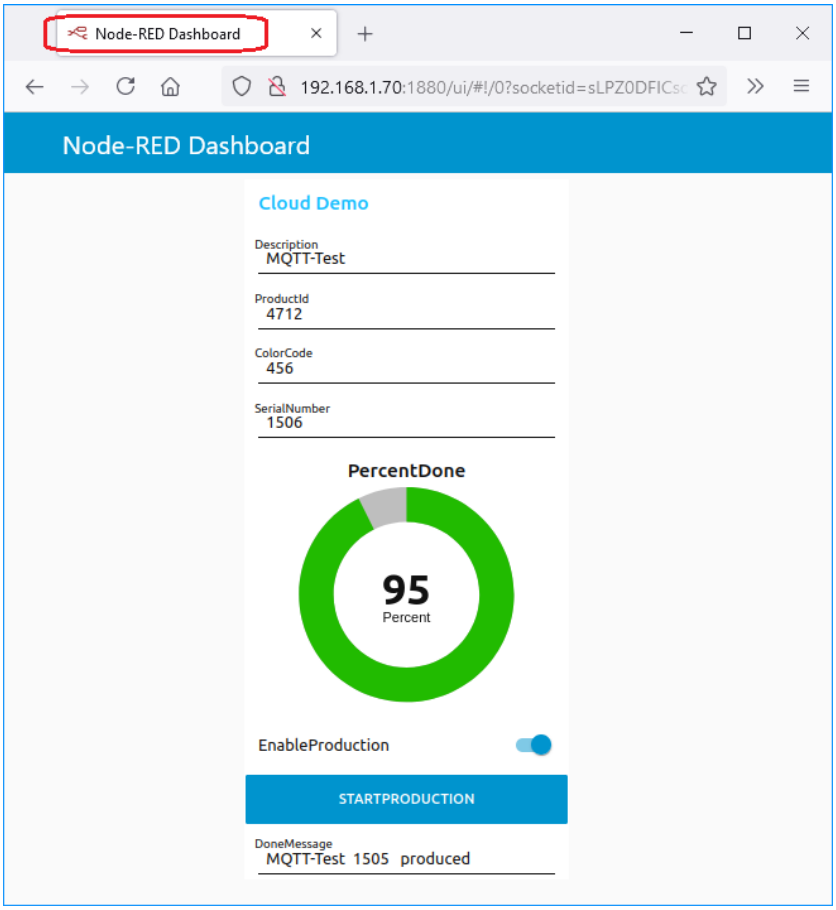
Weboberfläche

Die Weboberfläche kann nun ein einem beliebigen Browser oder auf einem beliebigen Smartphone angezeigt werden. Die Adresse für die Anzeige von Node-RED ist die ***Brokeradresse*** und dem Port ***1883***.

< 192.168.1.70:18880 >

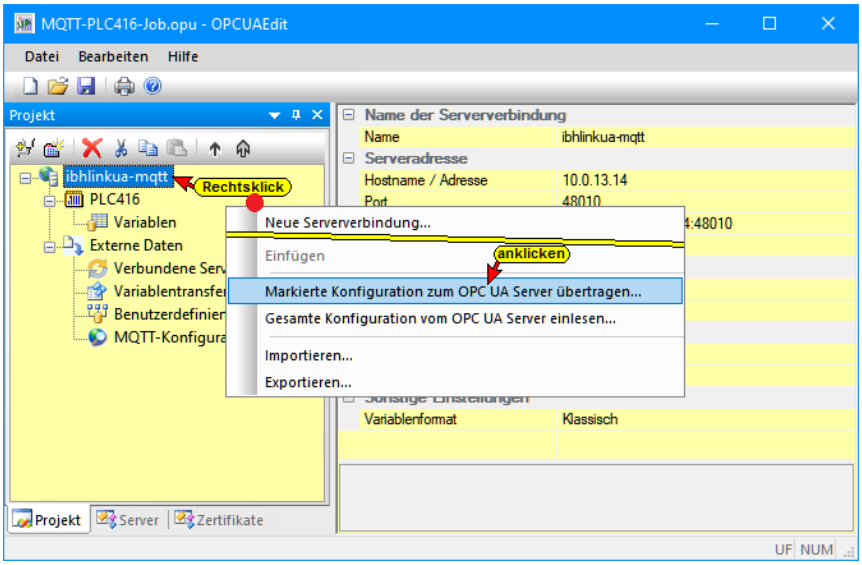
Um das **Node-RED Dashboard** anzuzeigen ist **/ui** dem Port hinzuzufügen. **< 192.168.1.70:18880/ui >**

Node-RED Dashboard



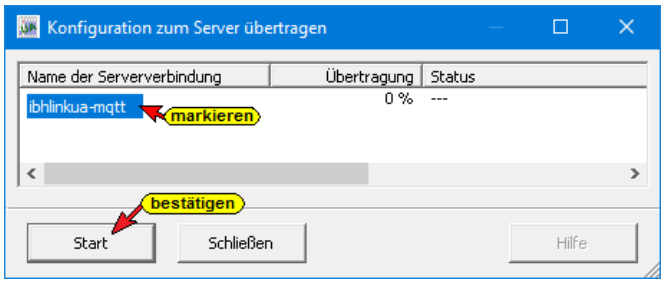
1.2.2 Konfiguration zum OPC UA Server übertragen

Sind alle Variablen definiert wird die Konfiguration an den IBH Link UA übertragen. Ein Rechtsklick auf das Symbol ***Server*** (IBH Link UA) öffnet das Kontextmenü.

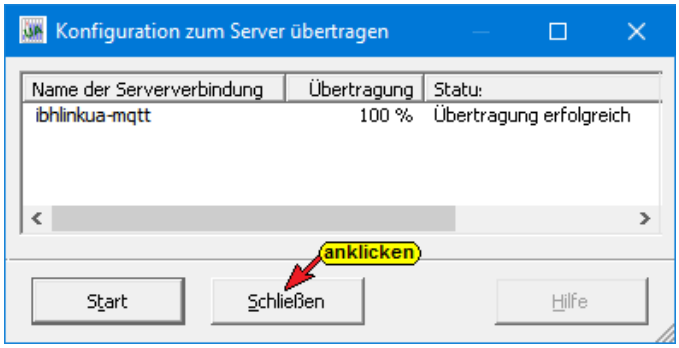


Der Befehl ***Markierte Konfiguration zum OPC UA Server übertragen*** öffnet das Dialogfeld Konfiguration zum Server übertragen.

Dialogfeld Konfiguration zum Server übertragen

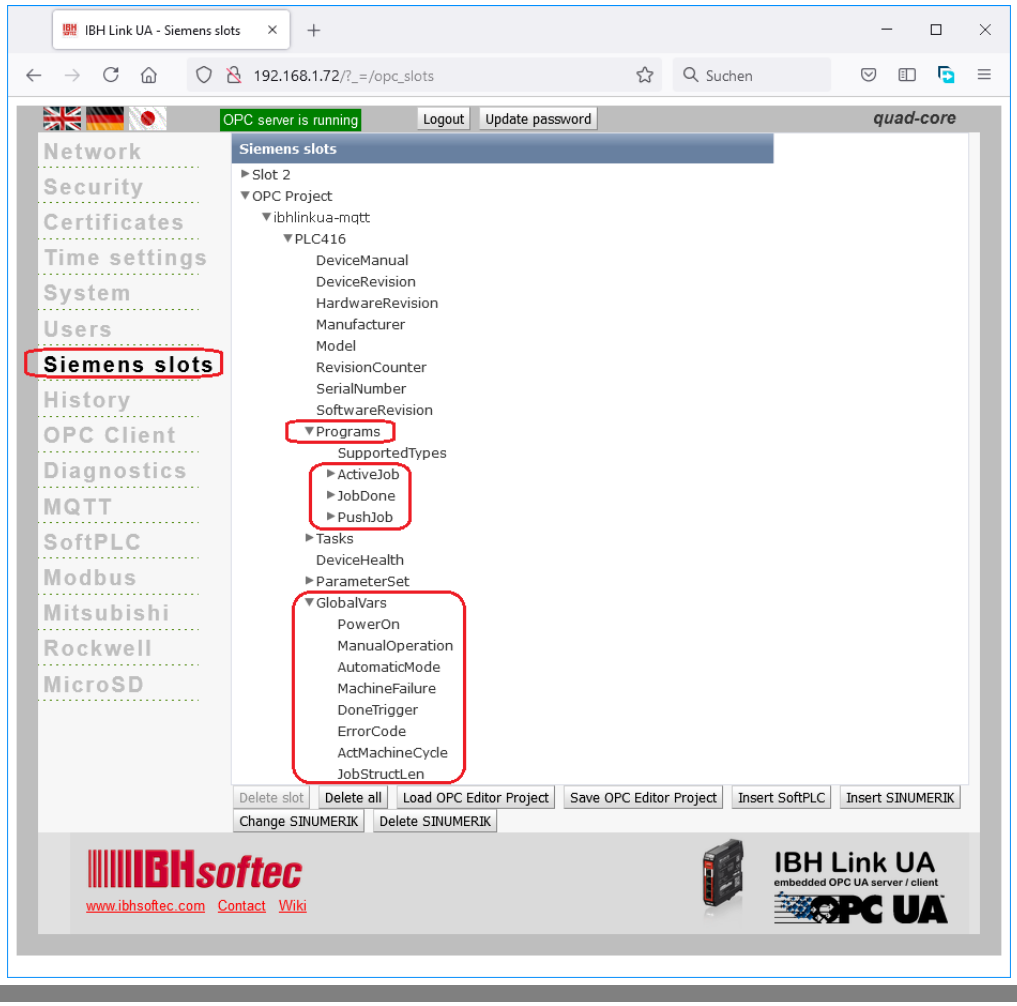


Mit markieren des Servers und anschließenden Anklicken von ***Start***, erfolgt die Übertragung.



Die erfolgreiche Übertragung wird angezeigt.

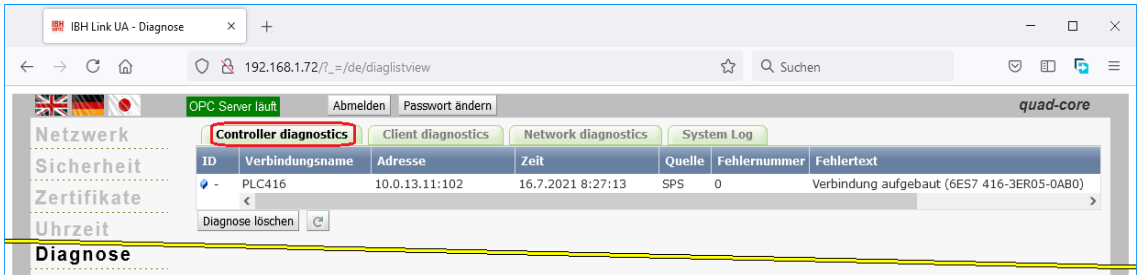
Aus dem OPC-Editor übernommene Informationen



IBH Link UA – Browser-Fenster Diagnose

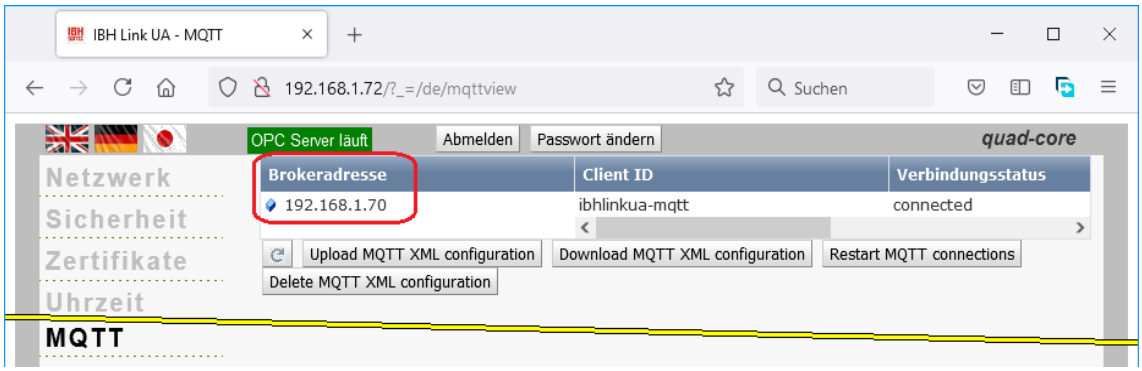
Steuerungsdiagnose

Es wird die konfigurierten Verbindungen und deren Status (fehlerfrei) zur CPU PLC416 angezeigt.



IBH Link UA – Browser-Fenster MQTT

Der Verbindungsstatus (connected) zum konfigurierten MQTT-Broker wird angezeigt.

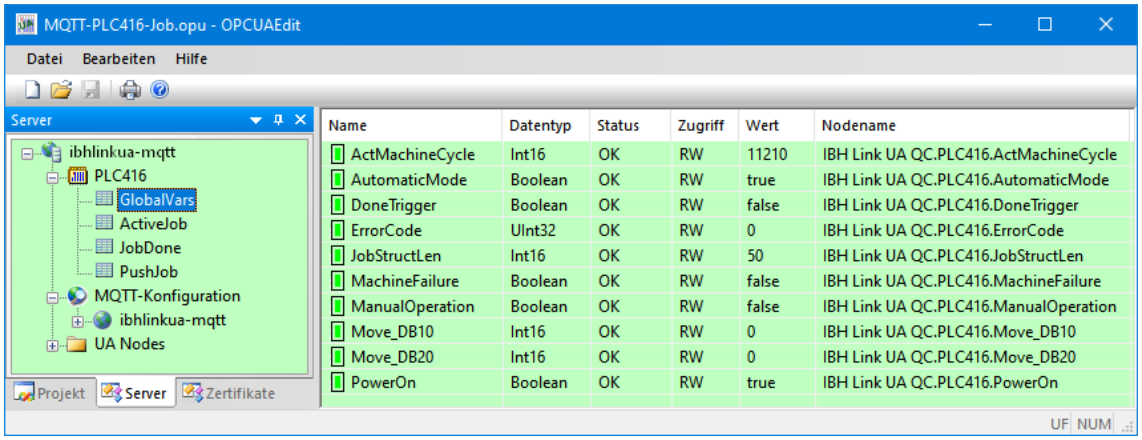


1.2.3 IBH OPC UA Editor Server-Fenster

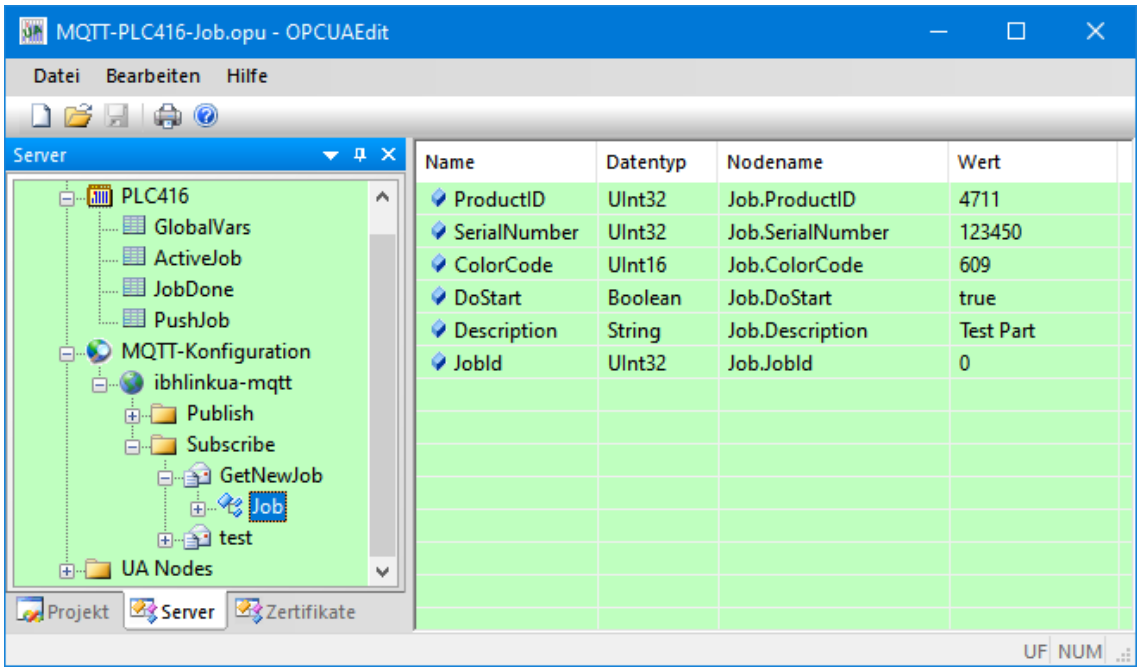
Die an den OPC UA Server übertragene Projektkonfiguration wird im Server-Fenster online angezeigt.

Es werden Informationen von dem ***online*** verbundenen ***OPC UA Server*** mit der online verbundenen ***CPU*** und der ***MQTT-Konfiguration*** angezeigt.

PLC416 online



MQTT online



UaExpert – Client

Mit ***Drag & Drop*** wurden MQTT-Information in das Fenster ***Data Access Viewer*** gezogen.