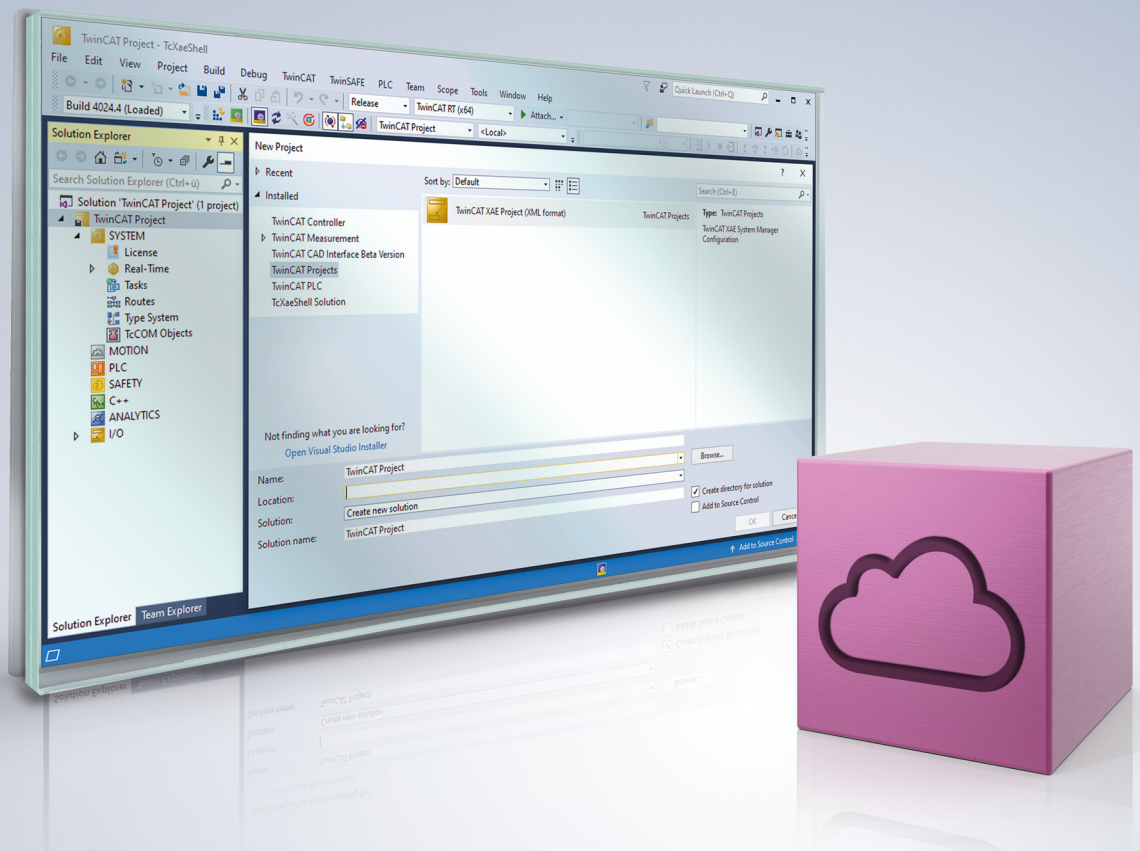
BECKHOFF New Automation Technology

Handbuch | DE

TF6730-TF6735

TwinCAT 3 | IoT Communicator (-App)



06.02.2024 | Version: 1.7.0

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

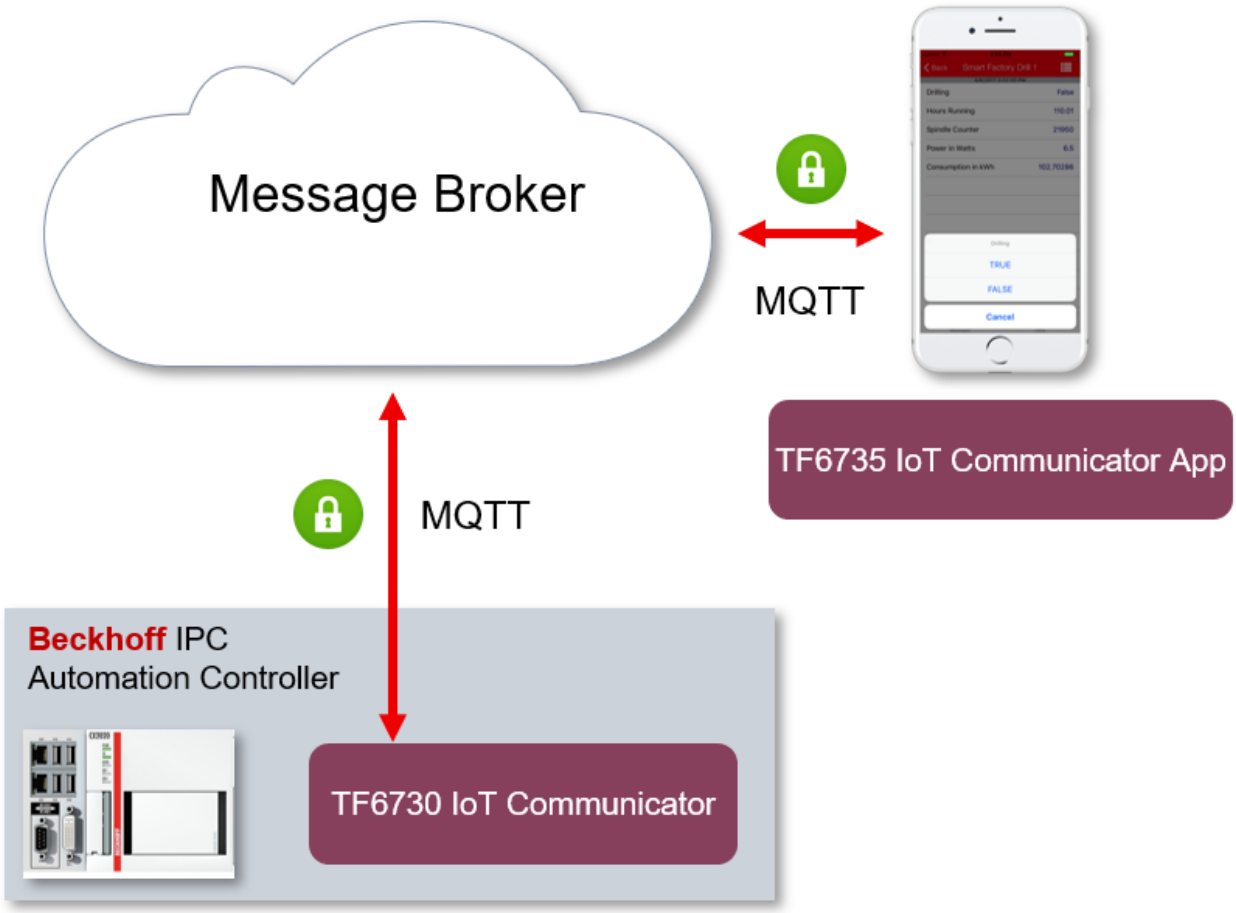
Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Mit den Funktionsbausteinen der SPS-Bibliothek Tc3_IoTCommunicator kann ein Datenaustausch zwischen der lokalen TwinCAT SPS und einem mobilem Endgerät (Smart Device) über einen MQTT-Message-Broker realisiert werden. Symbole können hierbei sowohl gesendet als auch empfangen werden. Nachrichten können auf dem Broker gespeichert und via Smart Device gelesen und gelöscht werden. Dazu muss auf dem mobilen Endgerät die TwinCAT IoT Communicator App installiert und ausgeführt werden.



Die TwinCAT IoT Communicator App steht im Apple AppStore und Google PlayStore kostenfrei zum Download zur Verfügung.

Download on the App Store

GET IT ON Google Play

Google Play and the Google Play logo are trademarks of Google Inc.

3 Installation

3.1 Systemvoraussetzungen

Technische Daten	Beschreibung
Betriebssystem	Windows 7/10, Windows Embedded Standard 7, Windows CE 7
Zielplattform	PC-Architektur (x86, x64 und ARM)
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4022.0 und höher
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAE, XAR
Erforderliche TwinCAT-Lizenz	TF6730 TC3 IoT Communicator
Einzubindende TwinCAT-Bibliothek	TC3_IotCommunicator

3.2 Installation

Für die Function TF6730 IoT Communicator wird kein separates Setup benötigt. Die benötigten Treiber-Komponenten werden direkt mit dem TwinCAT Setup (XAE und XAR) ausgeliefert.

3.3 Lizenzierung

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT-3-Lizenzierung“.

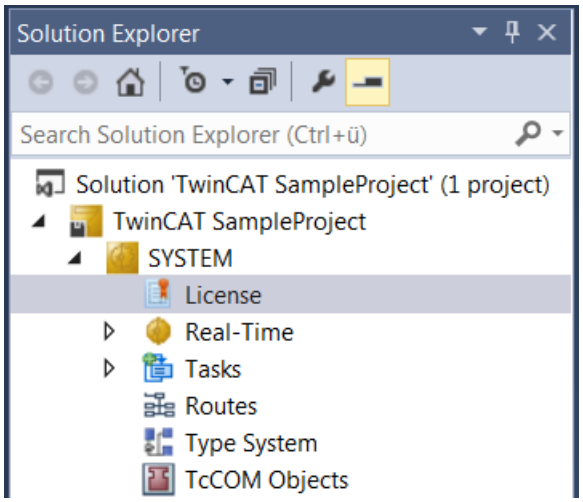
Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.

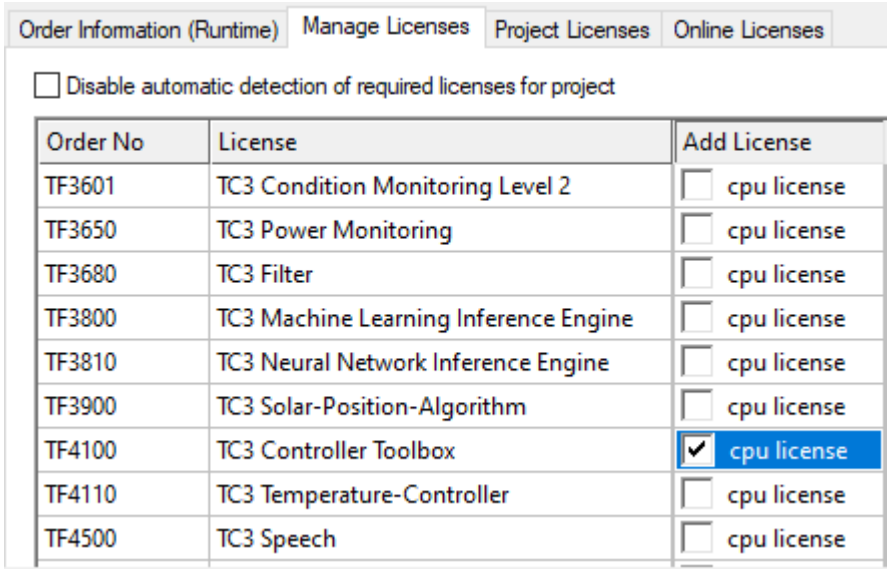
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.

4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**



⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.

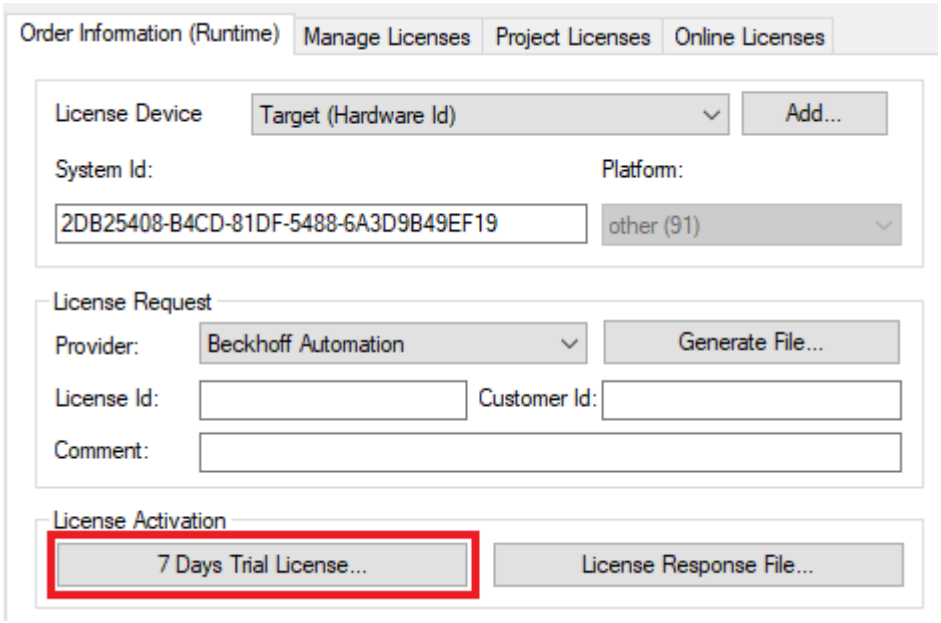
5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).



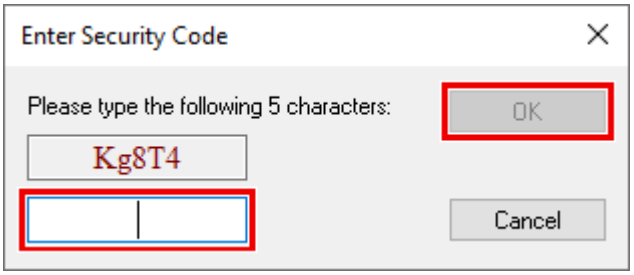
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt.

7. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



8. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
9. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
   ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
10. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

4 Technische Einführung

4.1 MQTT

4.1.1 Broker

Um Prozessdaten und Meldungen mit einem Smart Device auszutauschen bzw. zu synchronisieren, wird ein MQTT-Broker benötigt.

Der MQTT-Broker muss von der TwinCAT SPS und dem mobilem Endgerät via IP-Adresse oder Hostname erreichbar sein. TwinCAT und Smartphone müssen nicht direkt verbunden werden.

MQTT ist ein Publisher/Subscriber-basiertes Kommunikationsprotokoll, welches eine Nachrichten-basierte Übertragung zwischen Applikationen ermöglicht. Eine zentrale Komponente bei dieser Art der Übertragung ist der sogenannte Message Broker. Dieser hat die Aufgabe, Nachrichten zwischen den einzelnen Applikationen bzw. dem Sender und Empfänger einer Nachricht zu verteilen. Der Message Broker entkoppelt dabei Sender und Empfänger voneinander, sodass diese keine gegenseitigen Addressinformationen kennen und austauschen müssen. Alle Kommunikationsteilnehmer wenden sich beim Senden und Empfangen an den Message Broker und dieser übernimmt die Verteilung der Nachrichten.

MQTT-Broker Voraussetzungen für TC3 IoT Communicator

Für die optimale Nutzung der TwinCAT IoT Communicator App sollte der MQTT-Broker folgende Voraussetzungen erfüllen:

- MQTT-Protokollversion 3.1.1 (siehe Spezifikation OASIS-Standard)
- Clients benötigen den Zugang auf die Topics (siehe Topicstruktur)
- Retain-Nachrichten und Quality of Service 0 & 1 (siehe Quality of Service [▶ 12])

4.1.2 Quality of Service

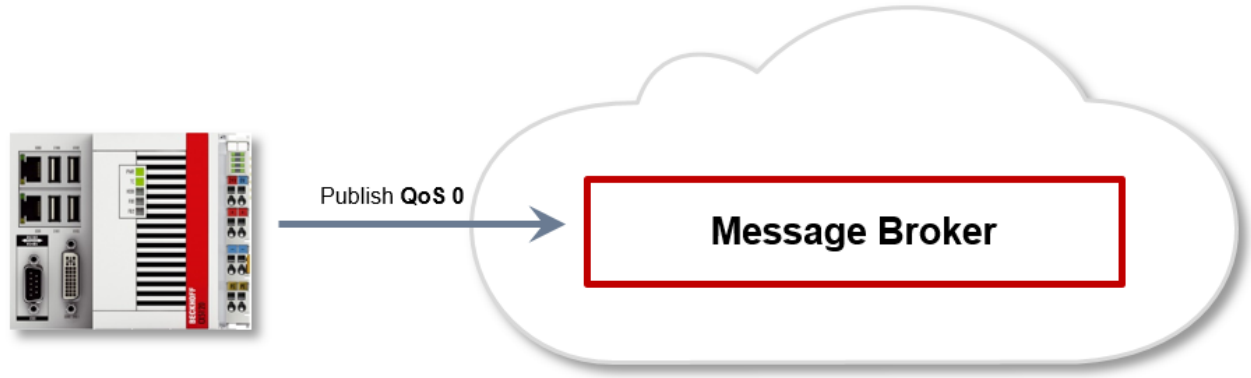
Quality of Service (QoS) ist eine Vereinbarung zwischen dem Sender und Empfänger einer Nachricht in Bezug auf das Garantieren der Nachrichtenübermittlung. Es existieren drei verschiedene Level in MQTT:

- 0 – höchstens einmal
- 1 – mindestens einmal
- 2 – genau einmal

Beide Kommunikationsarten (Publish/Subscribe) zum Message Broker müssen berücksichtigt und getrennt voneinander betrachtet werden. Das QoS-Level, welches ein Client beim Publishen einer Nachricht verwendet, wird vom jeweiligen Client gesetzt. Wenn der Broker nun die Nachricht an einen Client weiterleitet, der sich entsprechend auf das Topic subscribed hat, wird das QoS-Level vom Subscriber verwendet, welches beim Herstellen der Subscription angegeben wurde. Dies bedeutet, dass ein QoS-Level, welches vom Publisher vielleicht mit 2 angegeben wurde, vom Subscriber mit 0 „überschrieben“ werden kann.

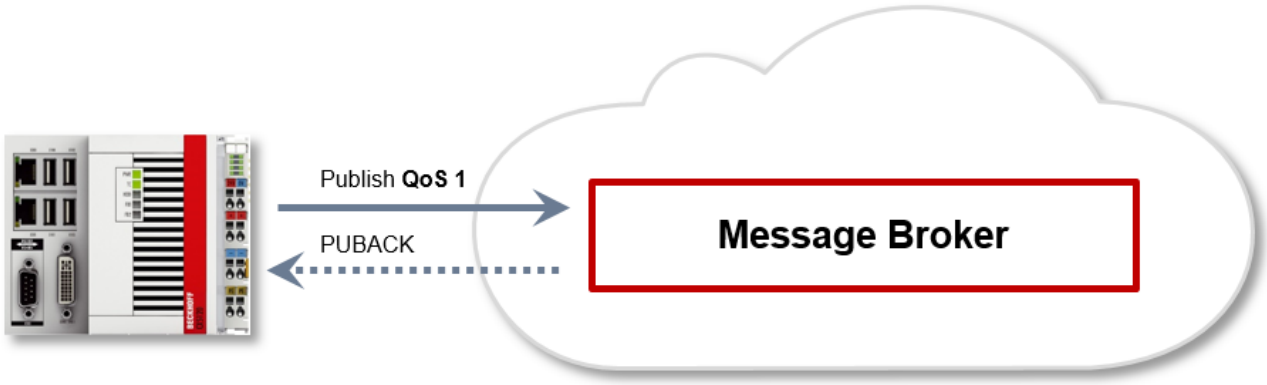
QoS-Level 0

Bei diesem QoS-Level erfolgt keine Bestätigung des Empfängers, ob die Nachrichten empfangen wurden oder nicht. In der Folge wird die Nachricht auch kein zweites Mal gesendet.



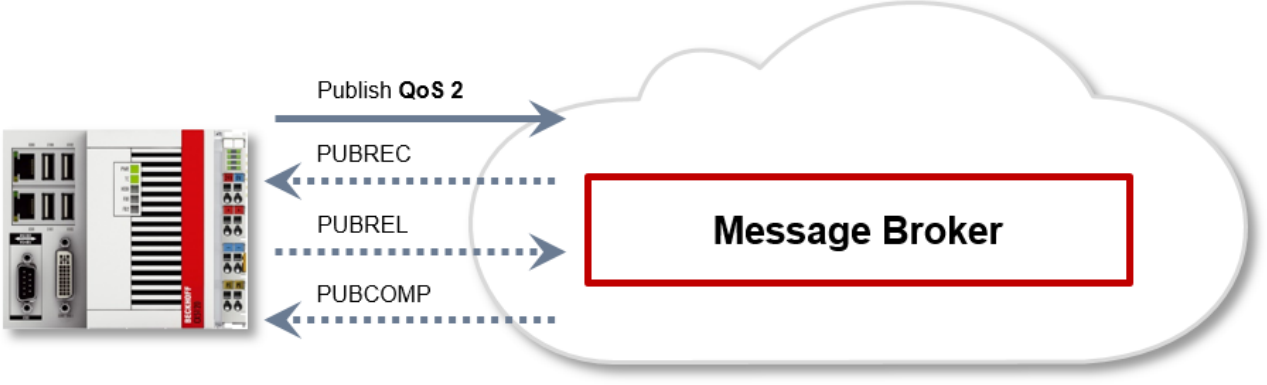
QoS-Level 1

Bei diesem QoS-Level wird garantiert, dass die Nachricht zumindest einmal beim Empfänger ankommt. Aber die Nachricht kann unter Umständen auch mehrfach beim Empfänger eintreffen. Der Sender speichert die Nachricht intern bis er eine Bestätigung in Form einer PUBACK-Nachricht vom Empfänger erhält. Wenn die PUBACK-Nachricht für eine bestimmte Zeit ausbleibt, wird die Nachricht erneut gesendet.



QoS-Level 2

Bei diesem QoS-Level wird garantiert, dass die Nachricht maximal einmal beim Empfänger ankommt. Dies wird MQTT-seitig durch einen Handshake-Mechanismus realisiert. QoS-Level 2 ist der sicherste (aus Sicht der Nachrichtenübermittlung), aber auch langsamste QoS-Level. Wenn ein Empfänger eine Nachricht mit QoS-Level 2 erhält, bestätigt er die Nachricht mit einem PUBREC. Der Absender der Nachricht merkt sich diese intern bis er ein PUBCOMP empfangen hat. Dieser zusätzliche Handshake (verglichen mit QoS 1) ist wichtig, damit die Nachricht nicht doppelt übertragen wird. Wenn der Absender der Nachricht ein PUBREC erhält, kann er den initiale Publish verwerfen, da er weiß, dass die Nachricht einmal vom Empfänger empfangen wurde. Er merkt sich somit intern den PUBREC und sendet seinerseits ein PUBREL. Nachdem der Empfänger ein PUBREL empfangen hat, kann er die sich zuvor gemerkten Zustände verwerfen und mit einem PUBCOMP antworten. Umgekehrt genauso. Immer dann, wenn ein Paket verloren geht, ist der jeweilige Kommunikationsteilnehmer dafür verantwortlich, die zuletzt gesendete Nachricht nach einer bestimmten Zeit noch einmal zu senden.



4.2 Sicherheit

Die MQTT-Spezifikation bietet einem MQTT-Client die Möglichkeit zur Benutzung von Username/Password-Authentifizierung am Message Broker. Zusätzlich können gängige Kryptographie-Mechanismen, wie z. B. TLS (Transport Layer Security), verwendet werden, um die Datenkommunikation zwischen Client und Message Broker zusätzlich abzusichern.

4.2.1 Authentifizierung

Die SPS-Bibliothek TC3_IoTCommunicator, sowie die TwinCAT IoT Communicator App können einen Authentifizierungsmechanismus nutzen, welcher im MQTT-Protokoll standardisiert und implementiert ist (siehe Spezifikation OASIS-Standard). Die SPS-Bibliothek und die App nutzen die MQTT-Protokollversion 3.1.1.

HINWEIS

Authentifizierung garantiert keinen Schutz vor Cyber-Attacken

Implementieren Sie zusätzlich zur Authentifizierung eine TLS-Verschlüsselung. Anderenfalls werden Benutzername und Passwort im Klartext übertragen. (Siehe Verschlüsselung [▶ 14])

4.2.2 Verschlüsselung

Eine Verschlüsselung und Authentifizierung via TLS kann mithilfe einer Zertifizierungsstelle (engl. Certificate Authority (CA)) erfolgen. Hierbei stellt die CA eine Signatur über den Public Key sowohl für den Message Broker (der sogenannte „Server-Key“) als auch üblicherweise für alle sich verbindenden Clients aus. Alle Kommunikationsteilnehmer können sich dann gegenseitig vertrauen, indem sie der ausstellenden Zertifizierungsstelle vertrauen. Je nach Message Broker kann sich ein MQTT Client jedoch auch ohne eigenes Client-Zertifikat verbinden. Hierbei verwendet der Client dann nur den Public Key der ausstellenden Zertifizierungsstelle während der Verbindungherstellung mit dem Broker.

5 Konfiguration

5.1 Attribute

Die in der Folge aufgelisteten Attribute sind die allgemein anwendbaren Attribute. Die speziell für die Widgets eingeführten Attribute werden unter Widgets [▶ 17] beschrieben.

Anzeigename der Variablen (iot.DisplayName)

Syntax: {attribute 'iot.DisplayName' := 'Ceiling Lights'}

Definiert den Namen, der in der App für diese Variable angezeigt werden soll. Wenn dieses Attribut nicht angegeben wird, wird in der App der SPS-Variablenname angezeigt.

Einheit der Variablen (iot.Unit)

Syntax: {attribute 'iot.Unit' := '°C'}

Definiert die Einheit, die in der App hinter dem Wert der Variablen steht. Wenn dieses Attribut nicht angegeben wird, bleibt die Einheit hinter dem Wert leer.

Variable nicht veränderbar (iot.Readonly)

Syntax: {attribute 'iot.ReadOnly' := 'TRUE'}

Definiert, ob die Variable von der App aus verändert werden darf. Wird dieses Attribut mit dem Wert TRUE angegeben, ist die Variable nicht mehr veränderbar und es erscheint ein Schloss-Symbol neben dem Variablennamen. Wenn dieses Attribut nicht angegeben wird, ist die Variable standardmäßig veränderbar.

Icon einer verschachtelten Struktur (iot.NestedStructIcon)

Syntax: {attribute 'iot.NestedStructIcon' := 'Room'}

Definiert das Icon für die Startseite einer verschachtelten Struktur. Standardmäßig wird die TwinCAT-CD angezeigt. Die verfügbaren Icons werden in der Liste der verfügbaren Icons [▶ 66] aufgelistet.

Minimal- und Maximalwert der Variable (iot.MinValue und iot.MaxValue)

Syntax: {attribute 'iot.MinValue' := '10'} {attribute 'iot.MaxValue' := '30'}

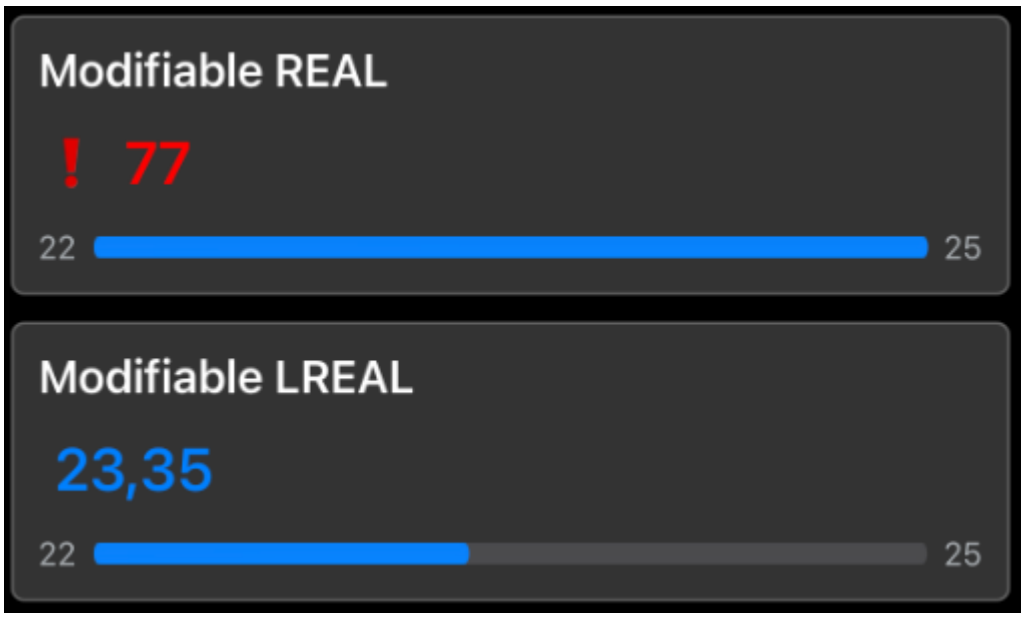
Definiert einen Minimal- und Maximalwert für numerische Variablen. Wenn beide Attribute ('MinValue' UND 'MaxValue') angegeben werden, wird in der App anhand einer Fortschrittsanzeige angezeigt, inwieweit der aktuelle Wert in Bezug auf den Minimal- und Maximalwert fortgeschritten ist.

HINWEIS

Fortschrittsanzeige

Minimal- und Maximalwert definieren den Bereich, den die Fortschrittsanzeige in der App berücksichtigt. Der Wert kann in der SPS die angegebenen Werte über- oder unterschreiten.

Wenn ein Wert seinen vorgeschriebenen Bereich verlässt, wird dies in der App mithilfe eines rot markierten Wertes hervorgehoben. Im folgenden Screenshot hat ein Wert seinen definierten Bereich verlassen.



Beschränkung der Dezimalstellen an einer Variablen (iot.DecimalPrecision)

Syntax: {attribute 'iot.DecimalPrecision' := '3'}

Definiert eine Anzahl an Nachkommastellen, auf die ein Gleitkommazahlwert gerundet wird. Diese Einstellung überschreibt für die jeweilige Variable eine eventuell vorhandene App-Einstellung.

Beispiel

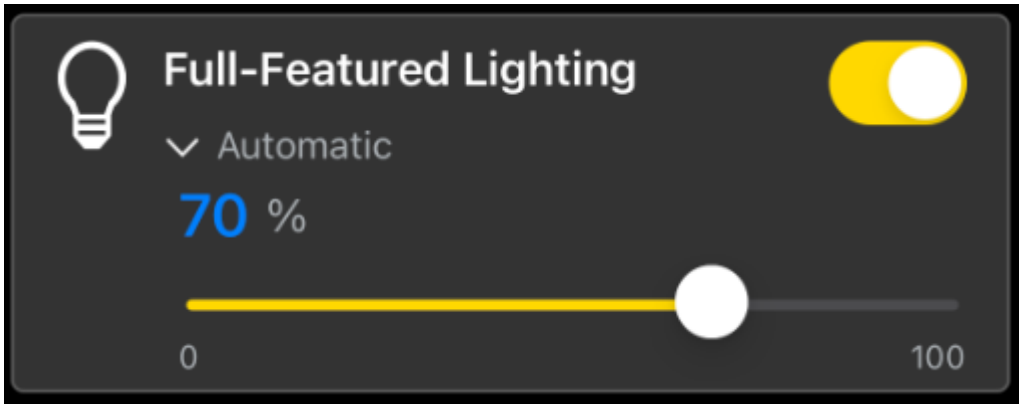
Siehe auch: Beispiele > Applikationsbeispiel [▶ 63]

5.2 Widgets

Mit den Widgets wird die App um Funktionen für die Bedienung einer Gebäudeautomatisierung erweitert. Die Widgets sind erstmals mit App-Version 1.4.0 und Tc3_IotCommunicator-Bibliotheks-Version 1.1.14.0 verfügbar und werden in folgenden Versionen um weitere Widgets erweitert.

5.2.1 Beleuchtung

Das beschriebene Widget eignet sich für die Darstellung von Leuchtmitteln in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'Lighting'}
{attribute 'iot.LightValueVisible' := 'true'}
{attribute 'iot.LightSliderVisible' := 'true'}
{attribute 'iot.LightModeVisible' := 'true'}
{attribute 'iot.LightModeChangeable' := 'true'}
stLightingWidgetSample : ST_LightingWidgetSample;
```

Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von *sDisplayName* überschrieben, sobald *sDisplayName* kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: *Lighting*.
iot.LightValueVisible	BOOL	Legt fest, ob der Dimmwert angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightSliderVisible	BOOL	Legt fest, ob der Slider angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

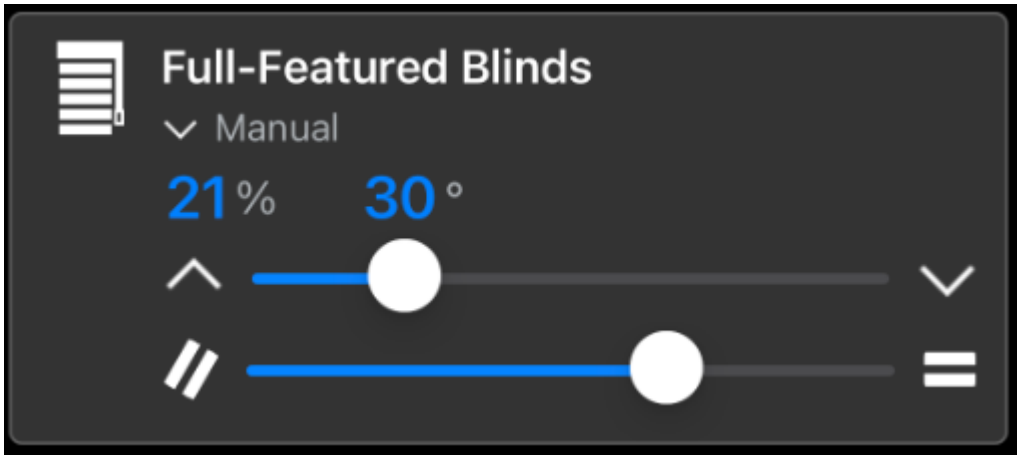
```
TYPE ST_LightingWidgetSample :
STRUCT
    sDisplayName    : STRING := '';
    bLight          : BOOL := FALSE;
    {attribute 'iot.Unit' := '%'}
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nLight          : INT := 100;
```

```
    sMode           : STRING := 'Automatic';
    aModes          : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bLight	BOOL	Schaltet die Beleuchtung an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
iot.Unit	STRING	Einheit des Dimmwertes.	Einheit hinter dem Zahlenwert.
iot.MinValue	INT	Untere Grenze des Dimmwertes.	Auf der linken Seite unter dem Slider.
iot.MaxValue	INT	Obere Grenze des Dimmwertes.	Auf der rechten Seite unter dem Slider.
nLight	INT	Dimmwert des Widgets.	Anzeige im Zahlenwert und zusätzlich Abbildung in der Füllung des Sliders.
sMode	STRING	Modus der Beleuchtung.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.2 Jalousien

Das beschriebene Widget eignet sich für die Darstellung von Jalousien in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'Blinds'}
{attribute 'iot.BlindsPositionValueVisible' := 'true'}
{attribute 'iot.BlindsPositionSliderVisible' := 'true'}
{attribute 'iot.BlindsAngleValueVisible' := 'true'}
{attribute 'iot.BlindsAngleSliderVisible' := 'true'}
{attribute 'iot.BlindsModeVisible' := 'true'}
{attribute 'iot.BlindsModeChangeable' := 'true'}
stBlindsWidgetSample : ST_BlindsWidgetSample;
```

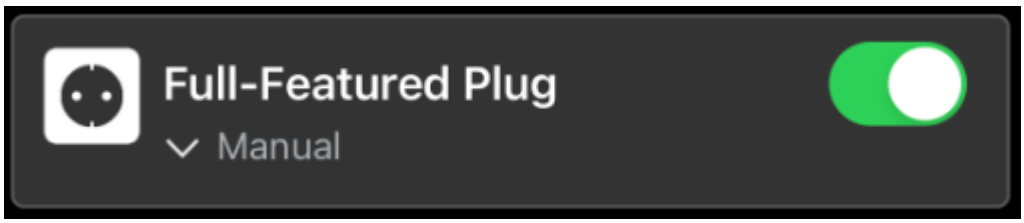
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder zusätzlich auch Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: Blinds.
iot.BlindsPositionValueVisible	BOOL	Legt fest, ob der Positionswert angezeigt wird (TRUE) oder nicht (FALSE).
iot.BlindsPositionSliderVisible	BOOL	Legt fest, ob der Slider für den Positionswert angezeigt wird (TRUE) oder nicht (FALSE).
iot.BlindsAngleValueVisible	BOOL	Legt fest, ob der Winkelwert angezeigt wird (TRUE) oder nicht (FALSE).
iot.BlindsAngleSliderVisible	BOOL	Legt fest, ob der Slider für den Winkelwert angezeigt wird (TRUE) oder nicht (FALSE).
iot.BlindsModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.BlindsModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

```
TYPE ST_BlindsWidgetSample :
STRUCT
    sDisplayName     : STRING := '';
    bActive : BOOL;
    bPositionUp      : BOOL;
    bPositionDown    : BOOL;
    bAngleUp         : BOOL;
    bAngleDown       : BOOL;
    {attribute 'iot.Unit' := '%'}
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nPositionValue   : INT;
    nPositionRequest : INT;
    {attribute 'iot.Unit' := '°'}
    {attribute 'iot.MinValue' := '-90'}
    {attribute 'iot.MaxValue' := '90'}
    nAngleValue      : INT;
    nAngleRequest    : INT;
    sMode            : STRING := 'Automatic';
    aModes           : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bActive	BOOL	Vorgesehen, um eine Aktivierung der Jalousien zu erkennen.	Klicken auf die Fläche des Widgets an Stellen, wo keine anderen Kontrollelemente sind.
bPositionUp	BOOL	Vorgesehen für das Hochfahren der Jalousien.	Button auf der linken Seite des oberen Sliders.
bPositionDown	BOOL	Vorgesehen für das Herunterfahren der Jalousien.	Button auf der rechten Seite des oberen Sliders.
bAngleUp	BOOL	Vorgesehen für das Stellen des Winkelwertes in Richtung des minimalen Wertes.	Button auf der linken Seite des unteren Sliders.
bAngleDown	BOOL	Vorgesehen für das Stellen des Winkelwertes in Richtung des maximalen Wertes.	Button auf der rechten Seite des unteren Sliders.
iot.Unit	STRING	Einheit des Positionswertes.	Einheit hinter dem ersten Zahlenwert.
iot.MinValue	INT	Untere Grenze des Positionswertes.	Nur in der SPS abgebildet.
iot.MaxValue	INT	Obere Grenze des Positionswertes.	Nur in der SPS abgebildet.
nPositionValue	INT	Positionswert der Jalousien.	Der erste der beiden Zahlenwerte.
nPositionRequest	INT	Zielwert der Position der Jalousien.	Der Wert, der im Moment des Loslassens des oberen Sliders an die App geschickt wird.
iot.Unit	STRING	Einheit des Winkelwertes.	Einheit hinter dem zweiten Zahlenwert.
iot.MinValue	INT	Untere Grenze des Winkelwertes.	Nur in der SPS abgebildet.
iot.MaxValue	INT	Obere Grenze des Winkelwertes.	Nur in der SPS abgebildet.
nAngleValue	INT	Winkelwert der Jalousien.	Der zweite der beiden Zahlenwerte.
nAngleRequest	INT	Zielwert der Position der Jalousien.	Der Wert, der im Moment des Loslassens des unteren Sliders an die App geschickt wird.
sMode	STRING	Modus der Jalousien.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.3 Steckdose

Das beschriebene Widget eignet sich für die Darstellung einer Steckdose in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'Plug'}
{attribute 'iot.PlugModeVisible' := 'true'}
{attribute 'iot.PlugModeChangeable' := 'true'}
stPlugWidgetSample : ST_PlugWidgetSample;
```

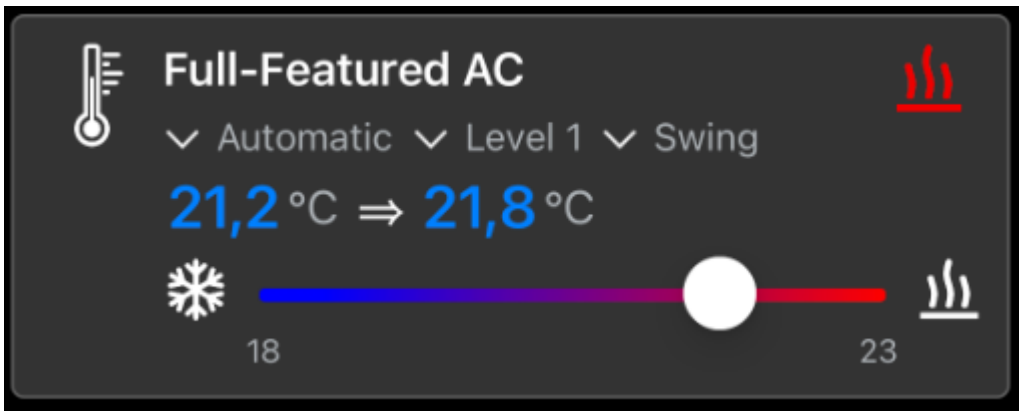
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von *sDisplayName* überschrieben, sobald *sDisplayName* kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: *Plug*.
iot.PlugModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.PlugModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

```
TYPE ST_PlugWidgetSample :
STRUCT
    sDisplayName    : STRING := '';
    bOn             : BOOL;
    sMode           : STRING := 'Automatic';
    aModes          : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bOn	BOOL	Schaltet die Steckdose an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
sMode	STRING	Modus der Steckdose.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.4 Klimaanlage

Das beschriebene Widget eignet sich für die Darstellung von Klimaanlagen in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'AC'}
{attribute 'iot.ACValueRequestVisible' := 'true'}
{attribute 'iot.ACSliderVisible' := 'true'}
{attribute 'iot.ACModeVisible' := 'true'}
{attribute 'iot.ACModeChangeable' := 'true'}
{attribute 'iot.ACModeStrengthVisible' := 'true'}
{attribute 'iot.ACModeStrengthChangeable' := 'true'}
{attribute 'iot.ACModeLamellaVisible' := 'true'}
{attribute 'iot.ACModeLamellaChangeable' := 'true'}
{attribute 'iot.DecimalPrecision' := '2'}
stACWidgetSample : ST_ACWidgetSample;
```

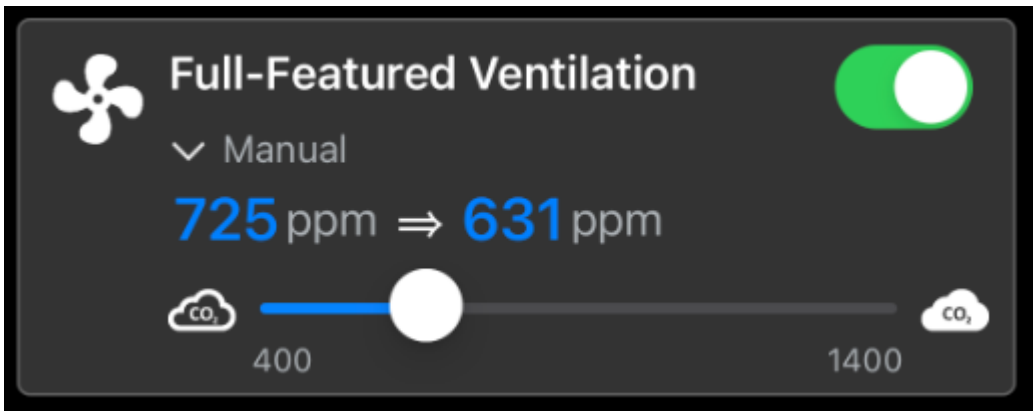
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: *AC*.
iot.ACValueRequestVisible	BOOL	Legt fest, ob der Zielwert hinter dem aktuellen Temperaturwert angezeigt wird (TRUE) oder nicht (FALSE).
iot.ACSliderVisible	BOOL	Legt fest, ob der Slider angezeigt wird (TRUE) oder nicht (FALSE).
iot.ACModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.ACModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).
iot.ACModeStrengthVisible	BOOL	Legt fest, ob der Modus für die Stärke angezeigt wird (TRUE) oder nicht (FALSE).
iot.ACModeStrengthChangeable	BOOL	Legt fest, ob der Modus für die Stärke einstellbar ist (TRUE) oder nicht (FALSE).
iot.ACModeLamellaVisible	BOOL	Legt fest, ob der Modus für die Lamellen angezeigt wird (TRUE) oder nicht (FALSE).
iot.ACModeLamellaChangeable	BOOL	Legt fest, ob der Modus für die Lamellen einstellbar ist (TRUE) oder nicht (FALSE).
iot.DecimalPrecision	INT	Legt die Anzahl nach Nachkommastellen fest. Diese Einstellung überschreibt die Einstellung an der Variable *nTemperature*.

```
TYPE ST_ACWidgetSample :
STRUCT
    sDisplayName        : STRING := '';
    nAcMode             : INT; // 0: Off, 1: Cooling, 2: Ventilating, 3: Heating, 4: Cooling Off, 5:
 Ventilating Off, 6: Heating Off
    {attribute 'iot.Unit' := '°C'}
    {attribute 'iot.MinValue' := '18'}
    {attribute 'iot.MaxValue' := '23'}
    {attribute 'iot.DecimalPrecision' := '2'}
    nTemperature        : LREAL;
    nTemperatureRequest : LREAL;
    sMode               : STRING := 'OnlyFromPLCMode';
    aModes              : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
    sMode_Strength      : STRING := 'Level 3';
    aModes_Strength     : ARRAY[0..2] OF STRING := ['Level 0', 'Level 1', 'Level 2'];
    sMode_Lamella       : STRING := 'QuickSwing';
    aModes_Lamella      : ARRAY[0..1] OF STRING := ['Static', 'Swing'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
nAcMode	INT	Legt den Modus des AC-Widgets fest.	Icon rechts oben. 0: kein Icon 1: Cooling (blau) 2: Ventilating (grün) 3: Heating (rot) 4: Cooling off (grau) 5: Ventilating off (grau) 6: Heating off (grau)
iot.Unit	STRING	Einheit des Temperaturwertes.	Einheit hinter beiden Zahlenwerten.
iot.MinValue	INT	Untere Grenze des Temperaturbereichs.	Auf der linken Seite unter dem Slider.
iot.MaxValue	INT	Obere Grenze des Temperaturbereichs.	Auf der rechten Seite unter dem Slider.
iot.DecimalPrecision	INT	Anzahl der Nachkommastellen für die Temperaturwerte. Wird von der DecimalPrecision am Widget überschrieben und gilt für beide Temperaturwerte.	Bei den beiden Temperaturwerten.
nTemperature	LREAL	Aktueller Temperaturwert.	Die Zahl auf der linken Seite des Pfeils.
nTemperatureRequest	LREAL	Angefragter Temperaturwert, über den Slider in 0,1er-Schritten möglich.	Die über den Slider angefragte Temperatur. Wird anschließend auf der rechten Seite des Pfeils dargestellt.
sMode	STRING	Modus der Klimaanlage.	Der aktuell angezeigte Modus (links).
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren allgemeinen Modi.	Durch Drücken auf den aktuellen Modus (links) können die einstellbaren Modi eingeblendet werden.
sMode_Strength	STRING	Modus der Stufe der Klimaanlage.	Der aktuell angezeigte Modus (mittig).
aModes_Strength	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Stufen-Modi.	Durch Drücken auf den aktuellen Modus (mittig) können die einstellbaren Modi eingeblendet werden.
sMode_Lamella	STRING	Modus der Lamellen der Klimaanlage.	Der aktuell angezeigte Modus (rechts).
aModes_Lamella	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Lamellen-Modi.	Durch Drücken auf den aktuellen Modus (rechts) können die einstellbaren Modi eingeblendet werden.

5.2.5 Lüftung

Das beschriebene Widget eignet sich für die Darstellung einer Lüftung in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'Ventilation'}
{attribute 'iot.VentilationValueRequestVisible' := 'true'}
{attribute 'iot.VentilationSliderVisible' := 'true'}
{attribute 'iot.VentilationModeVisible' := 'true'}
{attribute 'iot.VentilationModeChangeable' := 'true'}
stVentilationWidgetSample : ST_VentilationWidgetSample;
```

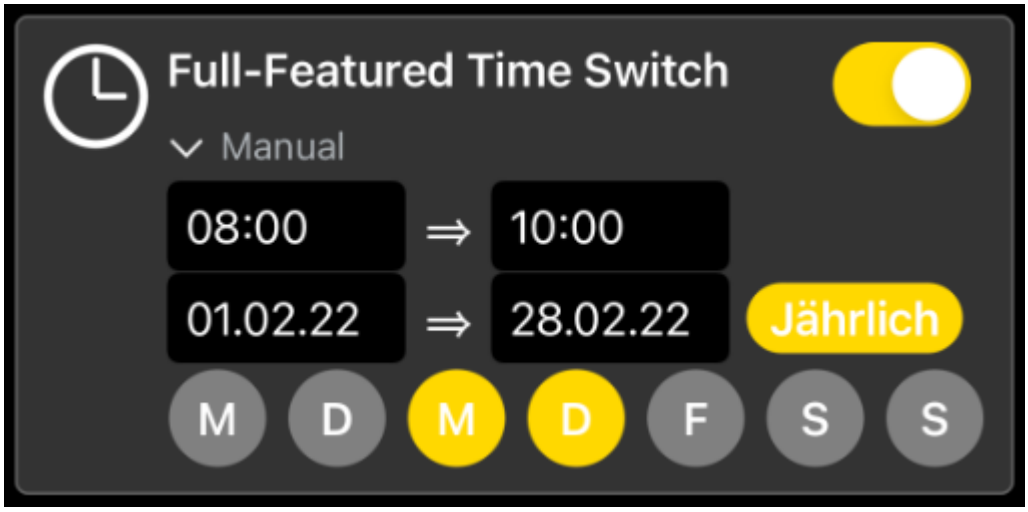
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: Ventilation.
iot.VentilationValueRequestVisible	BOOL	Legt fest, ob der angefragte Luftwert (CO2-Konzentration) angezeigt wird (TRUE) oder nicht (FALSE).
iot.VentilationSliderVisible	BOOL	Legt fest, ob der Slider angezeigt wird (TRUE) oder nicht (FALSE).
iot.VentilationModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.VentilationModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

```
TYPE ST_VentilationWidgetSample :
STRUCT
    sDisplayName    : STRING := '';
    bOn             : BOOL;
    {attribute 'iot.Unit' := 'ppm'}
    {attribute 'iot.MinValue' := '400'}
    {attribute 'iot.MaxValue' := '1400'}
    nValue          : INT;
    nValueRequest   : INT;
    sMode           : STRING := 'Automatic';
    aModes          : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bOn	BOOL	Schaltet die Lüftung an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
iot.Unit	STRING	Einheit des Luftwertes (der CO2-Konzentration).	Einheit hinter beiden Zahlenwerten.
iot.MinValue	INT	Untere Grenze des Luftwertes (der CO2-Konzentration).	Auf der linken Seite unter dem Slider.
iot.MaxValue	INT	Obere Grenze des Luftwertes (der CO2-Konzentration).	Auf der rechten Seite unter dem Slider.
nValue	INT	Aktueller Luftwert (CO2-Konzentration).	Die Zahl links vom Pfeil.
nValueRequest	INT	Angefragter Luftwert (CO2-Konzentration) über den Slider.	Die über den Slider angefragte Zahl. Wird anschließend auf der rechten Seite des Pfeils dargestellt.
sMode	STRING	Modus der Lüftung.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.6 Zeitschaltuhr

Das beschriebene Widget eignet sich für die Darstellung einer Zeitschaltuhr in der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'TimeSwitch'}
{attribute 'iot.TimeSwitchStartTimeVisible' := 'true'}
{attribute 'iot.TimeSwitchEndTimeVisible' := 'true'}
```

```
{attribute 'iot.TimeSwitchStartDateVisible' := 'true'}
{attribute 'iot.TimeSwitchEndDateVisible' := 'true'}
{attribute 'iot.TimeSwitchDaysVisible' := 'true'}
{attribute 'iot.TimeSwitchDateYearlyVisible' := 'true'}
{attribute 'iot.TimeSwitchModeVisible' := 'true'}
{attribute 'iot.TimeSwitchModeChangeable' := 'true'}
stTimeSwitchWidgetSample : ST_TimeSwitchWidgetSample;
```

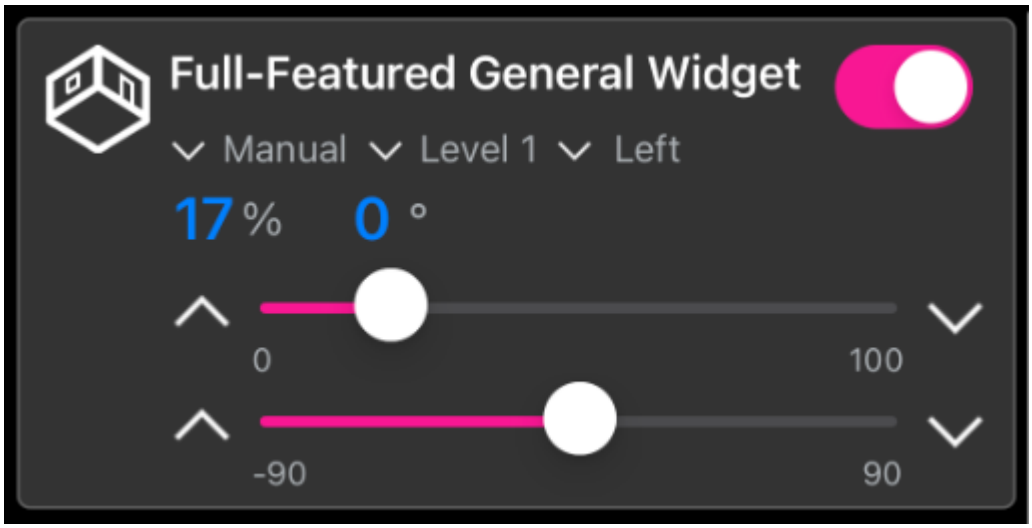
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: TimeSwitch.
iot.TimeSwitchStartTimeVisible	BOOL	Legt fest, ob die Startzeit angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchEndTimeVisible	BOOL	Legt fest, ob die Endzeit angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchStartDateVisible	BOOL	Legt fest, ob das Startdatum angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchEndDateVisible	BOOL	Legt fest, ob das Enddatum angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchDaysVisible	BOOL	Legt fest, ob die Wochentage angezeigt werden (TRUE) oder nicht (FALSE).
iot.TimeSwitchDateYearlyVisible	BOOL	Legt fest, ob das Attribut für die jährliche Konfiguration angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.TimeSwitchModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

```
TYPE ST_TimeSwitchWidgetSample :
STRUCT
    sDisplayName    : STRING := '';
    bOn             : BOOL;
    tStartTime      : TIME_OF_DAY;
    tEndTime        : TIME_OF_DAY;
    dStartDate      : DATE;
    dEndDate        : DATE;
    bYearly         : BOOL;
    bMonday         : BOOL;
    bTuesday        : BOOL;
    bWednesday      : BOOL;
    bThursday       : BOOL;
    bFriday         : BOOL;
    bSaturday       : BOOL;
    bSunday         : BOOL;
    sMode           : STRING := 'Automatic';
    aModes          : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bOn	BOOL	Schaltet die Zeitschaltuhr an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
tStartTime	TIME_OF_DAY	Startzeit der Zeitschaltuhr.	Uhrzeit auf der linken Seite des Pfeils.
tEndTime	TIME_OF_DAY	Endzeit der Zeitschaltuhr.	Uhrzeit auf der rechten Seite des Pfeils.
dStartDate	DATE	Startdatum der Zeitschaltuhr.	Datum auf der linken Seite des Pfeils.
dEndDate	DATE	Enddatum der Zeitschaltuhr.	Datum auf der rechten Seite des Pfeils.
bYearly	BOOL	Jährlich.	Jährlich/Yearly (Abhängig von der Sprache des Betriebssystems).
bMonday	BOOL	Montag.	M.
bTuesday	BOOL	Dienstag.	D.
bWednesday	BOOL	Mittwoch.	M.
bThursday	BOOL	Donnerstag.	D.
bFriday	BOOL	Freitag.	F.
bSaturday	BOOL	Samstag.	S.
bSunday	BOOL	Sonntag.	S.
sMode	STRING	Modus der Zeitschaltuhr.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.7 Generisches Widget

Das beschriebene Widget eignet sich für die Darstellung eines benutzerdefinierten Widgets in der App, das zusätzlich zu den spezifischen Widgets eine flexible Alternative darstellt. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'General'}
{attribute 'iot.GeneralWidgetIcon' := 'Room'}
{attribute 'iot.GeneralWidgetColor' := 'F81894'}
{attribute 'iot.GeneralValue1SwitchVisible' := 'true'}
{attribute 'iot.GeneralValue2Visible' := 'true'}
{attribute 'iot.GeneralValue2SliderVisible' := 'true'}
{attribute 'iot.GeneralValue2SliderValuesVisible' := 'true'}
{attribute 'iot.GeneralValue2SliderButtonsVisible' := 'true'}
{attribute 'iot.GeneralValue2SliderButtonsInverted' := 'true'}
{attribute 'iot.GeneralValue3Visible' := 'true'}
{attribute 'iot.GeneralValue3SliderVisible' := 'true'}
{attribute 'iot.GeneralValue3SliderValuesVisible' := 'true'}
{attribute 'iot.GeneralValue3SliderButtonsVisible' := 'true'}
{attribute 'iot.GeneralValue3SliderButtonsInverted' := 'true'}
{attribute 'iot.GeneralMode1Visible' := 'true'}
{attribute 'iot.GeneralMode1Changeable' := 'true'}
{attribute 'iot.GeneralMode2Visible' := 'true'}
{attribute 'iot.GeneralMode2Changeable' := 'true'}
{attribute 'iot.GeneralMode3Visible' := 'true'}
{attribute 'iot.GeneralMode3Changeable' := 'true'}
stGeneralWidgetSample : ST_GeneralWidgetSample;
```

Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: General.
iot.GeneralWidgetIcon	STRING	Legt fest, welches Icon für das Widget verwendet wird. Die Liste der verfügbaren Icons ist unter Liste der verfügbaren Icons [▶ 66] zu finden.
iot.GeneralWidgetColor	STRING	Legt fest, mit welcher Farbe Slider und Button angezeigt werden. Die Liste der verfügbaren Farben ist unter Liste der verfügbaren Farben [▶ 67] zu finden.
iot.GeneralValue1SwitchVisible	BOOL	Legt fest, ob der Button angezeigt wird (TRUE) oder nicht (FALSE).
iot.GeneralValue2Visible	BOOL	Legt fest, ob der linke der beiden Zahlenwerte sichtbar ist (TRUE) oder nicht (FALSE).
iot.GeneralValue2SliderVisible	BOOL	Legt fest, ob der obere der beiden Slider sichtbar ist (TRUE) oder nicht (FALSE).
iot.GeneralValue2SliderValuesVisible	BOOL	Legt fest, ob die Grenzen des oberen Sliders sichtbar sind (TRUE) oder nicht (FALSE).
iot.GeneralValue2SliderButtonsVisible	BOOL	Legt fest, ob die Buttons des oberen Sliders sichtbar sind (TRUE) oder nicht (FALSE).
iot.GeneralValue2SliderButtonsInverted	BOOL	Mit dieser Einstellung kann die Orientierung der Buttons des oberen Sliders umgedreht werden.
iot.GeneralValue3Visible	BOOL	Legt fest, ob der rechte der beiden Zahlenwerte sichtbar ist (TRUE) oder nicht (FALSE).
iot.GeneralValue3SliderVisible	BOOL	Legt fest, ob der untere der beiden Slider sichtbar ist (TRUE) oder nicht (FALSE).
iot.GeneralValue3SliderValuesVisible	BOOL	Legt fest, ob die Grenzen des unteren Sliders sichtbar sind (TRUE) oder nicht (FALSE).
iot.GeneralValue3SliderButtonsVisible	BOOL	Legt fest, ob die Buttons des unteren Sliders sichtbar sind (TRUE) oder nicht (FALSE).
iot.GeneralValue3SliderButtonsInverted	BOOL	Mit dieser Einstellung kann die Orientierung der Buttons des unteren Sliders umgedreht werden.
iot.GeneralMode1Visible	BOOL	Legt fest, ob der erste Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.GeneralMode1Changeable	BOOL	Legt fest, ob der erste Modus einstellbar ist (TRUE) oder nicht (FALSE).
iot.GeneralMode2Visible	BOOL	Legt fest, ob der zweite Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.GeneralMode2Changeable	BOOL	Legt fest, ob der zweite Modus einstellbar ist (TRUE) oder nicht (FALSE).
iot.GeneralMode3Visible	BOOL	Legt fest, ob der dritte Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.GeneralMode3Changeable	BOOL	Legt fest, ob der dritte Modus einstellbar ist (TRUE) oder nicht (FALSE).

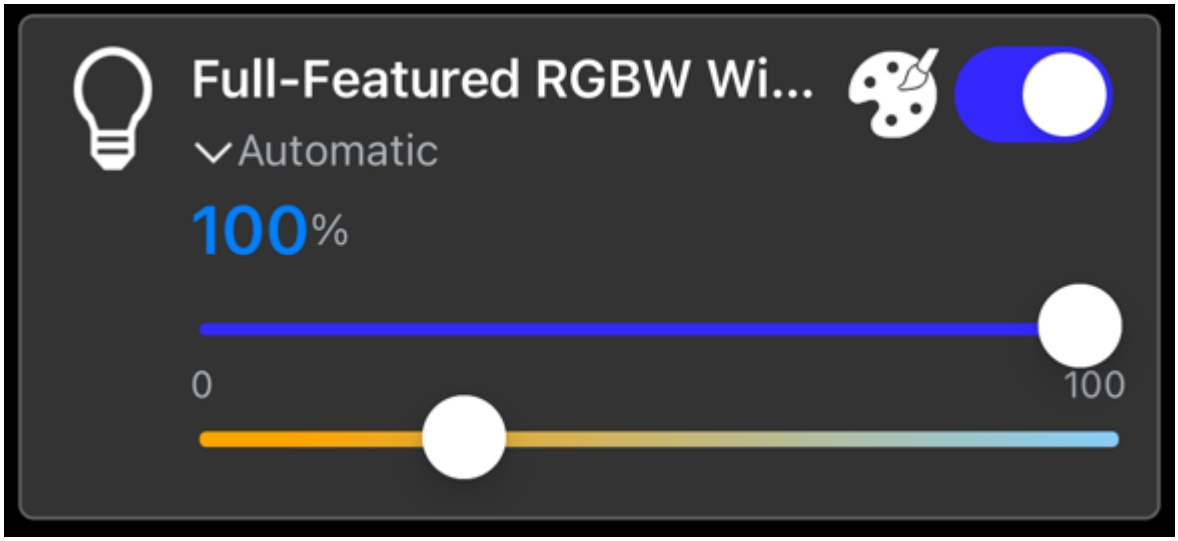
```
TYPE ST_GeneralWidgetSample :
STRUCT
    sDisplayName    : STRING := '';
    bValue1         : BOOL := FALSE;
    {attribute 'iot.Unit' := '%'}
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nValue2         : INT;
    nValue2Request  : INT;
    bValue2Up       : BOOL;
    bValue2Down     : BOOL;
    {attribute 'iot.Unit' := '%'}
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nValue3         : INT;
    nValue3Request  : INT;
    bValue3Up       : BOOL;
    bValue3Down     : BOOL;
    sMode1          : STRING := 'Automatic';
    aModes1         : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
    sMode2          : STRING := 'Automatic';
    aModes2         : ARRAY[0..2] OF STRING := ['Manual', 'Automatic', 'Next Mode'];
    sMode3          : STRING := 'Automatic';
    aModes3         : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bValue1	BOOL	Schaltet das Widget an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
iot.Unit	STRING	Einheit des ersten Wertes.	Einheit hinter dem ersten Zahlenwert.
iot.MinValue	INT	Untere Grenze des ersten Wertes.	Auf der linken Seite unter dem oberen Slider.
iot.MaxValue	INT	Obere Grenze des ersten Wertes.	Auf der rechten Seite unter dem oberen Slider.
nValue2	INT	Erster Wert.	Anzeige im linken Zahlenwert und zusätzlich Abbildung in der Füllung des oberen Sliders.
nValue2Request	INT	Anfragewert für den ersten Wert.	Der Wert, auf den der obere Slider bewegt wird.
nValue2Up	BOOL	Einer der Buttons für den oberen Slider.	Auf der linken Seite des oberen Sliders.
nValue2Down	BOOL	Einer der Buttons für den oberen Slider.	Auf der rechten Seite des oberen Sliders.
iot.Unit	STRING	Einheit des zweiten Wertes.	Einheit hinter dem zweiten Zahlenwert.
iot.MinValue	INT	Untere Grenze des zweiten Wertes.	Auf der linken Seite unter dem unteren Slider.
iot.MaxValue	INT	Obere Grenze des zweiten Wertes.	Auf der rechten Seite unter dem unteren Slider.
nValue3	INT	Zweiter Wert.	Anzeige im rechten Zahlenwert und zusätzlich Abbildung in der Füllung des unteren Sliders.
nValue3Request	INT	Anfragewert für den zweiten Wert.	Der Wert, auf den der untere Slider bewegt wird.
nValue3Up	BOOL	Einer der Buttons für den unteren Slider.	Auf der linken Seite des unteren Sliders.
nValue3Down	BOOL	Einer der Buttons für den unteren Slider.	Auf der rechten Seite des unteren Sliders.
sMode1	STRING	Erster Modus.	Der aktuell angezeigte erste Modus.
aModes1	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi für den ersten Modus.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.
sMode2	STRING	Zweiter Modus.	Der aktuell angezeigte zweite Modus.
aModes2	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi für den zweiten Modus.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.
sMode3	STRING	Dritter Modus.	Der aktuell angezeigte dritte Modus.

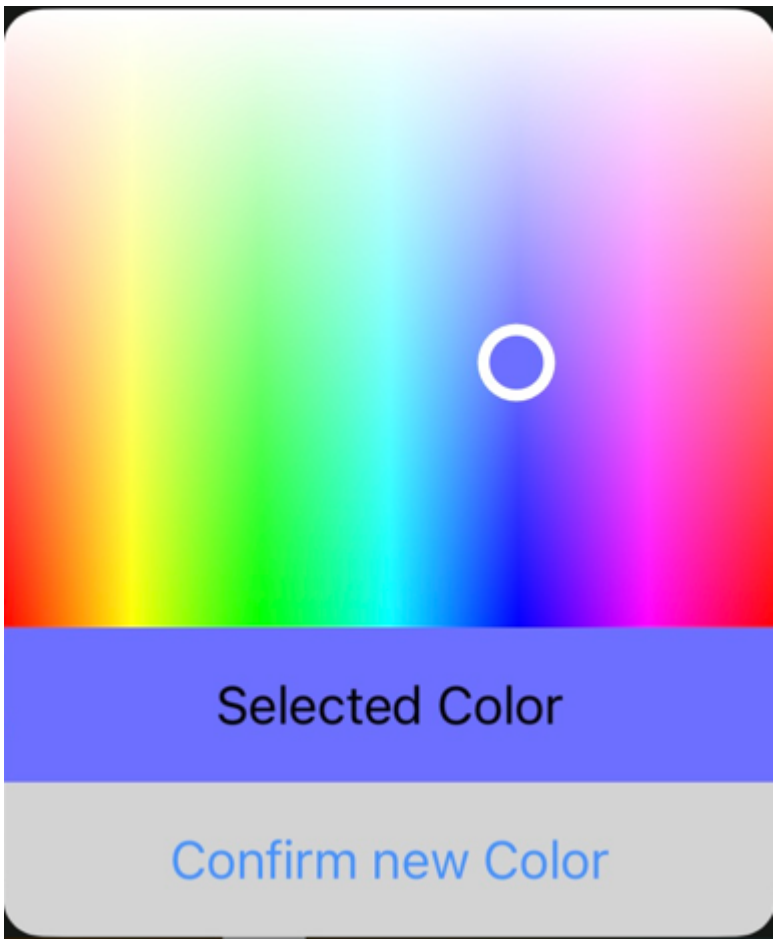
Attribut	Datentyp	Beschreibung	Abbildung im Widget
aModes3	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi für den dritten Modus.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.8 RGBW-Beleuchtung

Das beschriebene Widget eignet sich für die Bedienung von RGBW-Beleuchtungen aus der App. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Die folgende Abbildung zeigt die Farbpalette, die durch das Klicken auf das Farbpalettensymbol sichtbar wird.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'RGBW'}
{attribute 'iot.LightValueVisible' := 'true'}
{attribute 'iot.LightSliderVisible' := 'true'}
{attribute 'iot.LightColorPaletteVisible' := 'true'}
{attribute 'iot.LightColorTemperatureSliderVisible' := 'true'}
{attribute 'iot.LightModeVisible' := 'true'}
{attribute 'iot.LightModeChangeable' := 'true'}
stGeneralWidgetSample : ST_RGBWWidgetSample;
```

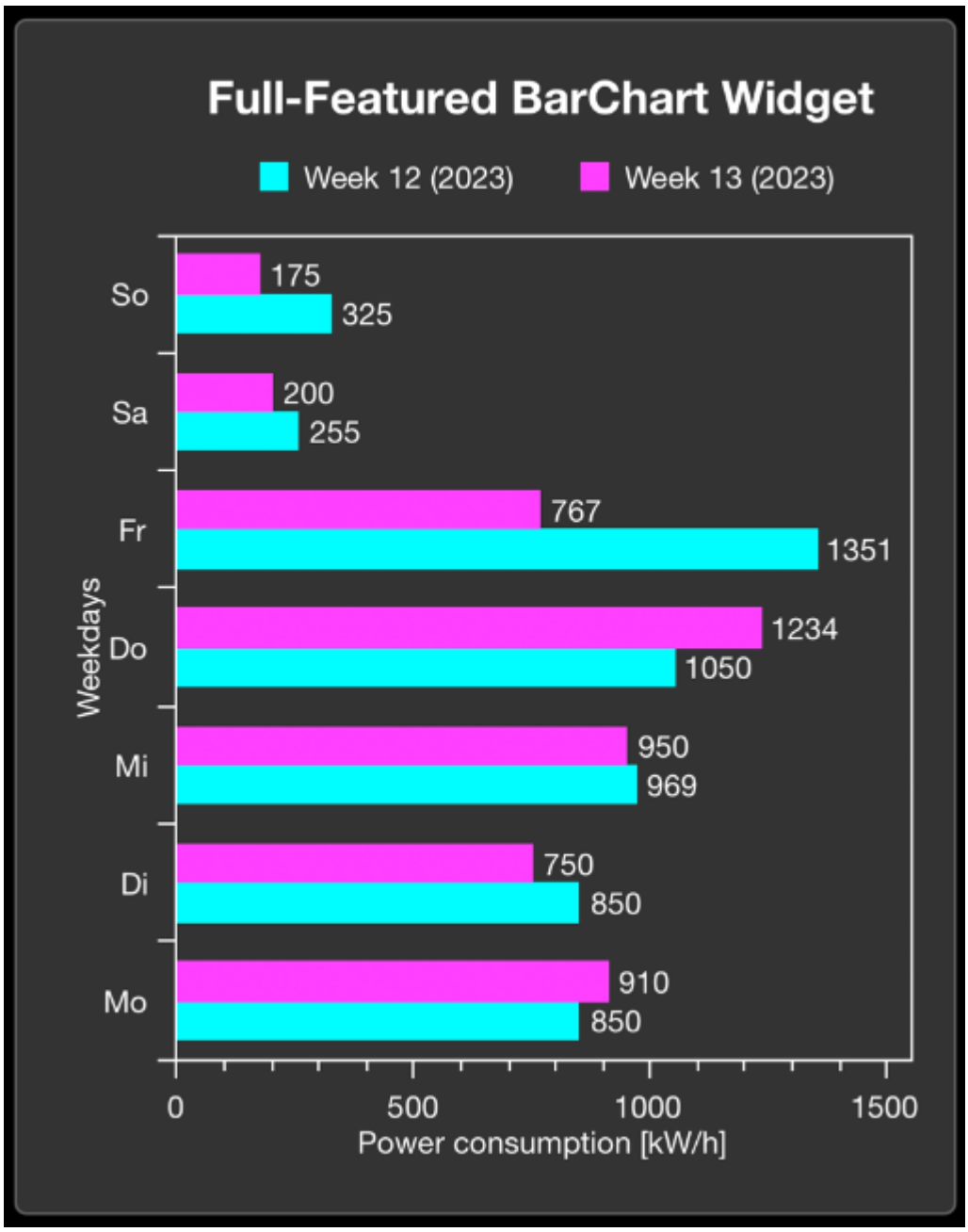
Attribut	Datentyp	Beschreibung
iot.ReadOnly	BOOL	Legt fest, ob das Widget auf Seite der App nur Lesezugriff (TRUE) oder auch zusätzlich Schreibzugriff in die SPS (FALSE) bekommt.
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: RGBW.
iot.LightValueVisible	BOOL	Legt fest, ob der Dimmwert angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightSliderVisible	BOOL	Legt fest, ob der obere der beiden Slider (für den Dimmwert des Lichts) sichtbar ist (TRUE) oder nicht (FALSE).
iot.LightColorPaletteVisible	BOOL	Legt fest, ob die Farbpalette angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightColorTemperatureSliderVisible	BOOL	Legt fest, ob der untere der beiden Slider (für die Farbtemperatur) sichtbar ist (TRUE) oder nicht (FALSE).
iot.LightModeVisible	BOOL	Legt fest, ob der Modus angezeigt wird (TRUE) oder nicht (FALSE).
iot.LightModeChangeable	BOOL	Legt fest, ob der Modus einstellbar ist (TRUE) oder nicht (FALSE).

```
TYPE ST_RGBWWidgetSample :
STRUCT
    sDisplayName      : STRING := '';
    bLight            : BOOL := FALSE;
    {attribute 'iot.Unit' := '%'}
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nLight            : INT := 100;
    nHueValue         : INT := 57;
    nSaturation       : INT := 100;
    {attribute 'iot.MinValue' := '2400'}
    {attribute 'iot.MaxValue' := '6500'}
    nColorTemperature : INT := 3500;
    sMode             : STRING := 'Automatic';
    aModes            : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
bLight	BOOL	Schaltet die Beleuchtung an (TRUE) oder aus (FALSE).	Kippschalter rechts oben.
iot.Unit	STRING	Einheit des Dimmwertes.	Einheit hinter dem Zahlenwert.
iot.MinValue	INT	Untere Grenze des Dimmwertes.	Auf der linken Seite unter dem oberen Slider.
iot.MaxValue	INT	Obere Grenze des Dimmwertes.	Auf der rechten Seite unter dem oberen Slider.
nLight	INT	Dimmwert der Beleuchtung.	Zahlenwert und zusätzlich Abbildung in der Füllung des oberen Sliders.
nHueValue	INT	Der Hue-Farbwert im Wertebereich von 0 (rot) bis 360 (wieder rot).	Die Gradanzahl des Kreises in der Farbpalette.
nSaturation	INT	Sättigung des Farbwertes im Wertebereich von 0 (grau) bis 100 (ausgewählte Farbe).	Ganz oben in der Farbpalette ist der Wert 0 und ganz unten in der Farbpalette ist der Wert 100.
iot.MinValue	INT	Untere Grenze des Farbtemperaturwertes.	Keine explizite Anzeige, definiert den Wertebereich des unteren Sliders.
iot.MaxValue	INT	Obere Grenze des Farbtemperaturwertes.	Keine explizite Anzeige, definiert den Wertebereich des unteren Sliders.
nColorTemperature	INT	Wert der Farbtemperatur.	Anzeige im unteren Slider.
sMode	STRING	Modus der Beleuchtung.	Der aktuell angezeigte Modus.
aModes	ARRAY [0..n] OF STRING	Array der verschiedenen durch den Benutzer einstellbaren Modi.	Durch Drücken auf den aktuellen Modus können die einstellbaren Modi eingeblendet werden.

5.2.9 Balkendiagramm

Das beschriebene Widget eignet sich für die Darstellung eines horizontalen Balkendiagramms in der App. Dieses Balkendiagramm kann sowohl einzelne Balken als auch zwei zu vergleichende Balken anzeigen. Die verschiedenen Konfigurationsmöglichkeiten werden im Folgenden beschrieben. In der Abbildung sind alle verfügbaren Features des Widgets aktiv.



Das Widget wird als Unterstruktur in der Gesamtstruktur der SendData [▶ 47]()-Methode übergeben. Für den Aufbau des Widgets werden bei der Deklaration der Struktur verschiedene SPS-Attribute verwendet.

```
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'BarChart'}
{attribute 'iot.ChartXAxisLabel' := 'Name for X Axis'}
{attribute 'iot.ChartYAxisLabel' := 'Name for Y Axis'}
{attribute 'iot.Unit' := 'Unit for X Axis'}
{attribute 'iot.ChartLegendVisible' := 'true'}
{attribute 'iot.ChartValuesVisible' := 'true'}
{attribute 'iot.MinValue':='0'}
{attribute 'iot.MaxValue':='1550'}
{attribute 'iot.ChartBarColor1' := '#00FFFF'}
{attribute 'iot.ChartBarColor2' := '#FF00FF'}
stChart : ST_BarChartWidgetSample;
```

Attribut	Datentyp	Beschreibung
iot.DisplayName	STRING	Der Anzeigename des Widgets in der App. Dieser wird von sDisplayName überschrieben, sobald sDisplayName kein Leerstring ist.
iot.WidgetType	STRING	Typangabe für das Widget, in diesem Fall: BarChart.
iot.ChartXAxisLabel	STRING	Der Anzeigename der X-Achse.
iot.ChartYAxisLabel	STRING	Der Anzeigename der Y-Achse.
iot.Unit	STRING	Einheit, die in eckigen Klammern hinter dem Anzeigenamen der X-Achse angezeigt wird.
iot.ChartLegendVisible	BOOL	Legt fest, ob die Legende angezeigt wird (TRUE) oder nicht (FALSE).
iot.ChartValuesVisible	BOOL	Legt fest, ob die Werte der Balken im Diagramm angezeigt werden (TRUE) oder nicht (FALSE).
iot.MinValue	INT	Untere Grenze des Wertebereichs des Diagramms.
iot.MaxValue	INT	Obere Grenze des Wertebereichs des Diagramms.
iot.ChartBarColor1	STRING	Legt die Farbe der oberen Balken fest. Die Liste der verfügbaren Farben ist unter Liste der verfügbaren Farben [▶ 67] zu finden.
iot.ChartBarColor2	STRING	Legt die Farbe der unteren Balken fest. Die Liste der verfügbaren Farben ist unter Liste der verfügbaren Farben [▶ 67] zu finden.

```
TYPE ST_BarChartWidgetSample :
STRUCT
    sDisplayName          : STRING := '';
    aDataSeries           : ARRAY[0..n] OF INT;
    aComparismDataSeries  : ARRAY[0..n] OF INT;
    aDataSeriesIdentifier : ARRAY[0..n] OF STRING;
    aLegendLabels         : ARRAY[0..n] OF STRING;
END_STRUCT
END_TYPE
```

Attribut	Datentyp	Beschreibung	Abbildung im Widget
sDisplayName	STRING	Legt den Anzeigenamen des Widgets fest und überschreibt das SPS-Attribut 'iot.DisplayName'.	Anzeigetext des Widgets.
aDataSeries	ARRAY [0..n] OF INT	Die erste Datenreihe, die im Balkendiagramm angezeigt wird. Das Array muss genauso groß sein wie die ComparismDataSeries und die DataSeriesIdentifier..	Länge der oberen Balken.
aComparismDataSeries	ARRAY [0..n] OF INT	Die zweite Datenreihe, die im Balkendiagramm angezeigt wird. Das Array muss genauso groß sein wie die DataSeries und die DataSeriesIdentifier. Beim Schicken eines leeren Arrays wird ein Balkendiagramm mit nur einem Balken angezeigt.	Länge der unteren Balken.
aDataSeriesIdentifier	ARRAY [0..n] OF STRING	Benennungen der Werte auf der Y-Achse. Dieses Array muss genauso groß sein wie die DataSeries und die ComparismDataSeries.	Beschriftungen der Werte auf der Y-Achse.
aLegendLabels	ARRAY [0..0] OF STRING ARRAY [0..1] OF STRING	Legende des Balkendiagramms. Die Größe des Arrays bestimmt sich dadurch, welche Anzahl an Balken dargestellt wird.	Zwischen dem Titel des Widgets und dem Diagramm.

5.3 Verschachtelte Strukturen

Es gibt die Möglichkeit, über mehrere Ebenen verschachtelte Strukturen aus der SPS in die Communicator-App zu kommunizieren. Diese verschachtelten Strukturen sind auf App-Seite direkt anzeigbar und können bis zur letzten Ebene ausgeklappt werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17 oder höher	App-Version 1.2.2 oder höher	Tc3_IotCommunicator

5.4 Beschränkung von Dezimalstellen

In vielen Anwendungsfällen reicht es in der App aus, nicht alle Nachkommastellen von Gleitkommazahlen (SPS: REAL und LREAL) anzuzeigen. Als Beispiel sei an dieser Stelle ein Temperaturwert genannt, bei dem ein Mensch maximal mit zwei Stellen nach dem Komma noch etwas anfangen kann.

An dieser Stelle gibt es zwei Möglichkeiten. die Anzahl an angezeigten Nachkommastellen zu beeinflussen. Bei der ersten Möglichkeit wird eine Einstellung für die gesamte App gesetzt, jede Variable wird auf eine in den App-Einstellungen festgelegte Anzahl an Nachkommastellen beschränkt (vgl. App-Einstellungen [▶ 59]). Die zweite Möglichkeit ist, über die SPS-Attribute eine bestimmte Anzahl an Nachkommastellen für eine einzelne Variable einzustellen (vgl. Beschränkung von Dezimalstellen [▶ 16]).

Wenn in den App-Einstellungen und der Einstellung an einer einzelnen Variablen verschiedene Werte für die Anzahl an Nachkommastellen definiert sind, wird die Einstellung an der einzelnen Variablen immer zuerst berücksichtigt. Es ist also möglich, beispielsweise für alle Gleitkommazahlen über die App-Einstellungen den Wert 2 zu definieren und trotzdem bei einzelnen Variablen von dieser Anzahl abzuweichen.

Die Beschränkung von Nachkommastellen bedeutet, dass die Werte gerundet werden. Sie werden keinesfalls abgeschnitten. Die untenstehende Tabelle zeigt ein einfaches Beispiel:

Wert	Decimal Number Precision	Anzeige in der App
1.68678	3	1.687
1.68678	1	1.7
1.68678	0	2

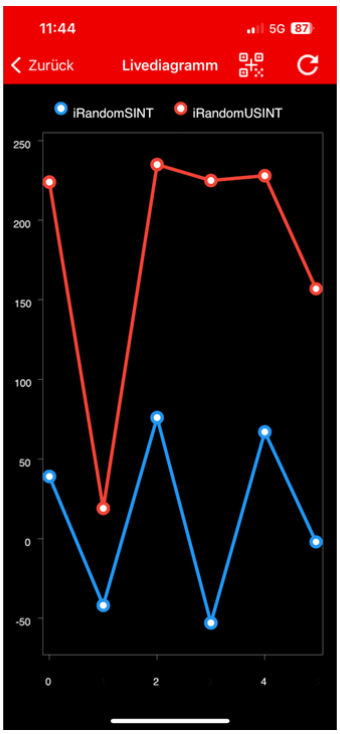
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.23 oder höher	App-Version 1.2.6 oder höher	Tc3_IotCommunicator

5.5 Navigation per QR-Code

Für jede Seite unterhalb eines Devices lässt sich ein QR-Code erzeugen. Mithilfe dieses QR-Codes ist es möglich, eine beliebige Seite der App direkt zu öffnen. Dafür kann zum einen der QR-Code als Grafik verwendet werden oder auch direkt die dahinter liegende URL.

Auch für einen Graphen lässt sich ein QR-Code erzeugen. Mit diesem QR-Code lässt sich dieser Graph ebenso wie eine einzelne Seite wieder öffnen. Dazu wird oben in der Navigationsleiste ein Button für das Erzeugen eines QR-Codes angezeigt.



5.6 OnChange-Mechanismen

Der OnChange-Mechanismus ist vorrangig für die Verwendung mit dem SendDataAsString [▶ 48]-Mechanismus vorgesehen. Es gibt verschiedene Hinweise zu der Nutzung dieses Mechanismus.

Unterschied der verschiedenen Methoden

Die SendData [▶ 47]-Methode und die SendDataAsString [▶ 48]-Methode werden als Retain-Nachrichten gesendet. Das hat den Effekt, dass eine neu verbundene App automatisch die zuletzt gesendeten Daten zur Verfügung hat, auch wenn gerade keine Daten gesendet werden.

Wenn die jeweiligen OnChange-Methoden verwendet werden, werden die Nachrichten nicht als Retain-Nachrichten gesendet. Bei einer neuen Verbindung sollte der aktuelle Stand der Daten als Retain-Nachricht gesendet werden, damit auch die neu verbundene App-Instanz den aktuellen Stand der Daten zur Verfügung hat. Weitere Informationen zu den OnChange-Methoden können unter SendData_OnChange [▶ 49] und SendDataAsString_OnChange [▶ 50] entnommen werden.

Hinzufügen von Variablen/Widgets

Wenn der SendDataAsString [▶ 48]-Mechanismus verwendet wird, muss ein bestimmter Aufbau des JSON-Dokuments berücksichtigt werden. Wenn zwischen zwei Aufrufen Änderungen (Hinzufügen/Löschen oder Anpassen) am Aufbau gemacht werden, sind folgende Punkte zu beachten:

- Wird eine Variable/ein Widget hinzugefügt und es ist keine Änderung der Reihenfolge gewünscht, kann der Parameter ForceUpdate auf FALSE bleiben.
- Wird eine Variable/ein Widget hinzugefügt und es ist eine Änderung der Reihenfolge gewünscht, muss der Parameter ForceUpdate für einen Aufruf auf TRUE gesetzt werden. Anschließend sollte er wieder auf FALSE gesetzt werden.
- Wird eine Variable/ein Widget gelöscht, muss die Seite in der App einmal geschlossen und wieder geöffnet werden.
- Wird eine Variable/ein Widget in Bezug auf die Metadaten verändert (bspw. Umkonfiguration eines Widgets) muss ForceUpdate für einen Aufruf auf TRUE gesetzt werden, ansonsten werden nur die Werte des Widgets aktualisiert. Anschließend sollte er wieder auf FALSE gesetzt werden.

Weitere Hinweise

- Der Parameter bNewAppSubscribe des FB_IotCommunicator [▶ 44]-Bausteins geht für einen Zyklus auf TRUE, wenn sich eine neue App-Instanz mit dem Topic auf dem Message Broker verbindet. Durch diesen Parameter kann der richtige Zeitpunkt zum Senden einer Retain-Nachricht identifziert werden.
- Der Parameter nActiveAppInstances des FB_IotCommunicator [▶ 44]-Bausteins gibt die Anzahl an verbundenen App-Instanzen an. Wenn keine App verbunden ist, kann auf das Senden von Daten verzichtet werden, um Performance und Datenmenge einzusparen.

Aufbau TwinCAT JSON

Die TwinCAT IoT Communicator-Produktpalette benutzt für die Kommunikation ein JSON-Format mit dem Namen TwinCAT JSON: Im Folgenden wird der Aufbau eines TwinCAT JSON-Dokuments am Beispiel der Widgets Steckdose [▶ 20] und Lüftung [▶ 25] beschrieben.

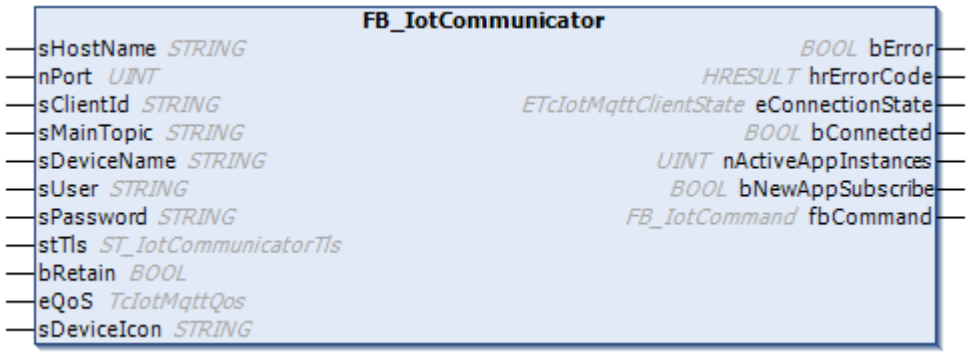
```
{
    "Timestamp" : "2022-08-04T07:15:06.176",
    "GroupName" : "Widget Testpage",
    "Values" : {
        "sPageDesc" : "TwinCAT JSON Page",
        "stPlug" : {
            "sDisplayName" : "",
            "bOn" : true,
            "sMode" : "Manual",
            "aModes" : [ "Manual", "Automatic" ]
        },
        "stVent" : {
             "sDisplayName" : "",
             "bOn" : true,
             "nValue" : 725,
             "nValueRequest" : 400,
             "sMode" : "Manual",
             "aModes" : [ "Manual", "Automatic" ]
        }
    },
    "MetaData" : {
        "sPageDesc" : {
            "iot.DisplayName" : "Info",
            "iot.ReadOnly" : "true"
        },
        "stPlug" : {
            "iot.DisplayName" : "Plug Widget",
            "iot.ReadOnly" : "false",
            "iot.WidgetType" : "Plug",
            "iot.PlugModeVisible" : "true",
            "iot.PlugModeChangeable" : "false"
        },
        "stVent" : {
            "iot.DisplayName" : "Ventilation Widget",
            "iot.ReadOnly" : "false",
            "iot.WidgetType" : "Ventilation",
            "iot.VentilationSliderVisible" : "true",
            "iot.VentilationValueRequestVisible" : "false",
            "iot.VentilationModeVisible" : "true",
            "iot.VentilationModeChangeable" : "false"
         },
        "stVent.nValue" : {
            "iot.Unit" : "ppm",
            "iot.MinValue" : "400",
            "iot.MaxValue" : "1400"
        }
    },
"ForceUpdate":false
}
```

Bereich	Beschreibung
Timestamp	Muss pro Nachricht einen Timestamp im Format: "YYYY-MM-DDThh:mm:ss.fff" enthalten z. B. "2022-08-04T07:15:06.176".
GroupName	Name des Einstiegsknotens des Communicator-Bausteins in der App.
Values	Die anzuzeigenden Werte, angefangen auf der ersten Seite, mit anschließender Verschachtelung.
MetaData	Alles, was in SPS-Attributen umgesetzt ist (bspw. die Konfiguration der Widgets).
ForceUpdate	Optionaler Parameter. Wird beim OnChange-Mechanismus dafür eingesetzt, nach Änderungen ein Update zu triggern. Genauere Informationen unter OnChange-Mechanismen [▶ 41].

6 SPS API

6.1 Funktionsbausteine

6.1.1 FB_IotCommunicator



Der Funktionsbaustein ermöglicht die Kommunikation zu einem MQTT-Broker.

Ein FB_IotCommunicator-Funktionsbaustein ist für die Verbindung zu genau einem Broker und für das Senden und Empfangen von Daten genau eines Gerätes („Device“) zuständig. Um die Hintergrundkommunikation zu diesem Broker zu gewährleisten und somit das Senden und Empfangen von Daten und Nachrichten zu ermöglichen, muss die Execute-Methode des Funktionsbausteins zyklisch aufgerufen werden.

Alle Verbindungsparameter sind als Eingangsparameter vorhanden und werden beim Verbindungsaufbau ausgewertet.

Syntax

Definition:

```
FUNCTION_BLOCK FB_IotCommunicator
VAR_INPUT
    sHostName        : STRING := '127.0.0.1'; // IP address/hostname of MQTT broker, default
'localhost'
    nPort            : UINT := 1883; // (*optional*) Port to use for connection to broker - default:
1883
    sClientId        : STRING; // (*optional*) Unique name to connect to MQTT broker
    sMainTopic       : STRING; // MQTT topic the TC3 IoT Communicator will communicate with
    sDeviceName      : STRING; // e.g. 'Machine XYZ' or 'Room 015 - Kitchen'
    sUser            : STRING; // (*optional*) Username for authentication
    sPassword        : STRING; // (*optional*) Password for specified User
    stTls            : ST_IotCommunicatorTls; // (*optional*) Specify details for secure tls
connection
    bRetain          : BOOL := TRUE; // (*optional*) default true - if false, data, messages &
devices will not be stored
    eQoS             : TcIotMqttQos := TcIotMqttQos.AtLeastOnceDelivery; // (*optional*) quality of
service between client & broker - default: 1
    sDeviceIcon      : STRING;
END_VAR
VAR_OUTPUT
    bError            : BOOL;
    hrErrorCode       : HRESULT;
    eConnectionState  : ETcIotMqttClientState;
    bConnected        : BOOL;
nActiveAppInstances : UINT;
bNewAppSubscribe : BOOL;
    fbCommand         : FB_IoTCommand; //provides functionality to receive commands
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sHostName	STRING	sHostName kann als Hostname oder als IP-Adresse angegeben werden. Bei fehlender Angabe wird Local Host verwendet.
nPort	UINT	Hier wird der Host Port angegeben. (Default: 1883)
sClientId	STRING	Die Client-ID kann individuell angegeben werden. Bei fehlender Angabe wird eine ID generiert.
sMainTopic	STRING	Hier wird angegeben, in welches Haupt-Topic die Daten und Nachrichten gesendet werden.
sDeviceName	STRING	Hier wird der Name des Gerätes eingetragen, zu dem die Daten und Nachrichten gehören.
sUser	STRING	Optional kann ein Benutzername angegeben werden.
sPassword	STRING	Zum Benutzernamen kann hier ein Passwort angegeben werden.
stTLS	STRING	Wenn der Broker eine TLS gesicherte Verbindung anbietet, kann hier die nötige Konfiguration vorgenommen werden. Die Parameterstruktur ist vom Typ ST_IotCommunicatorTls [▶ 52].
bRetain	BOOL	Die aktuellen Daten und die letzten 255 Nachrichten, sowie der aktuelle Status des Gerätes werden standardmäßig vom Broker gespeichert. Ist dies nicht erwünscht, kann bRetain auf FALSE gesetzt werden.
eQoS	TcIotMqttQos	Mit dieser Einstellung kann der Quality of Service (kurz: QoS) eingestellt werden.
sDeviceIcon	STRING	Mit dieser Einstellung kann das Icon des Communicator-Bausteins verändert werden. Wenn die Einstellung nicht gesetzt wird, wird standardmäßig die TwinCAT-CD als Icon verwendet. Die Liste der verfügbaren Icons ist unter Liste der verfügbaren Icons [▶ 66] zu finden.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, sobald eine Fehlersituation eintritt.
hrErrorCode	HRESULT	Liefert bei einem gesetzten bError-Ausgang einen Fehlercode.
eConnectionState	ETcIotMqttClientState	Gibt den Zustand der Verbindung vom Client zum Broker als Enumeration ETcIotMqttClientState an.
bConnected	BOOL	TRUE, wenn eine Verbindung vom Client zum Broker besteht.
nActiveAppInstances	UINT	Zeigt die Anzahl der aktuell verbundenen App-Instanzen.
bNewAppSubscribe	BOOL	Geht für einen Zyklus auf TRUE, wenn sich eine neue App-Instanz verbunden hat. Kann bei der Verwendung von OnChange-Funktionalitäten genutzt werden, um bei der Verbindung den aktuellen Stand der Daten wieder als Retain-Nachricht zu veröffentlichen.
fbCommand	FB_IoTCommand	Stellt alle nötigen Funktionalitäten bereit, um empfangene Daten („Commands“) auswerten zu können. Dieser Ausgang ist vom Typ FB_IotCommand [▶ 51].

Methoden

Name	Beschreibung
Execute [▶ 46]	Methode zur Hintergrundkommunikation mit dem TwinCAT-Treiber. Diese Methode muss zyklisch aufgerufen werden.
SendData [▶ 47]	Methode zum Senden von Daten an den angegebenen MQTT Message Broker als Retain-Nachricht.
SendMessage [▶ 47]	Methode zum Senden einer (Push-)Nachricht an den angegebenen MQTT Message Broker.
SendDataAsString [▶ 48]	Methode zum Senden von Daten an den angegebenen MQTT Message Broker bei der das JSON-Dokument direkt übergeben wird.
SendData_OnChange [▶ 49]	Methode zum Senden von OnChange-Daten an den angegebenen MQTT Message Broker.
SendDataAsString_OnChange [▶ 50]	Methode zum Senden von OnChange-Daten an den angegebenen MQTT Message Broker bei der das JSON-Dokument direkt übergeben wird.

Strings im UTF-8-Format

Die hier verwendeten Variablen vom Typ STRING nutzen das UTF-8-Format. Diese STRING-Formatierung ist üblich bei der MQTT-Kommunikation.

Um auch Sonderzeichen und Texte verschiedenster Sprachen empfangen zu können, wird der Zeichensatz in der Tc3_IotCommunicator-Bibliothek nicht auf den typischen Zeichensatz vom Datentyp STRING beschränkt. Stattdessen wird der Unicode-Zeichensatz als UTF-8-Format in Verbindung mit dem Datentyp STRING verwendet.

Bei Verwendung des ASCII-Zeichensatzes besteht kein Unterschied zwischen der typischen Formatierung in einem STRING und der UTF-8-Formatierung eines STRING.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.0	IPC oder CX (x86, x64, ARM)	Tc3_IotCommunicator

6.1.1.1 Execute

Diese Methode muss zyklisch aufgerufen werden, um die Hintergrundkommunikation mit dem MQTT Broker zu gewährleisten.

Syntax

```
METHOD Execute
VAR_INPUT
    bConnect : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bConnect	BOOL	Wenn bConnect auf TRUE gesetzt wird, findet der Verbindungsaufbau zum Broker statt. Um die Verbindung zu halten, muss bConnect gesetzt bleiben. Bei Aufruf der Execute()-Methode mit FALSE als Input, wird die Verbindung zum Broker abgebaut.

Mögliche Fehler werden an den Ausgängen bError, hrErrorCode und eConnectionState der Bausteininstanz ausgegeben

6.1.1.2 SendData

Diese Methode wird einmalig aufgerufen, um Daten an den Broker zu senden.

Syntax

```
METHOD SendData : BOOL
VAR_INPUT
    pMachineStruct : PVOID;
    nStructSize    : UINT;
END_VAR
```

Rückgabewert

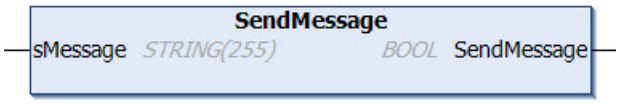
Name	Typ	Beschreibung
SendData	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
pMachineStruct	PVOID	Adresse zur Struktur, in der die Variablen des Gerätes deklariert werden.
nStructSize	UINT	Größe der bei pMachineStruct angegebenen Struktur.

Mögliche Fehler werden an den Ausgängen bError und hrErrorCode der Bausteininstanz ausgegeben.

6.1.1.3 SendMessage



Diese Methode wird einmalig aufgerufen, um eine (Push-)Nachricht an den Broker zu senden. Diese Nachricht wird dann in der App angezeigt. Es handelt sich hierbei nicht um eine Nachricht, die als Push-Mitteilung auf dem Handy sichtbar wird.

Syntax

```
METHOD SendMessage : BOOL
VAR_INPUT
    sMessage : STRING(255);
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
SendMessage	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
sMessage	STRING	Text der (Push-)Nachricht, die an den Broker geschickt werden soll.

Mögliche Fehler werden an den Ausgängen bError und hrErrorCode der Bausteininstanz ausgegeben.

Strings im UTF-8-Format

Die hier verwendeten Variablen vom Typ STRING nutzen das UTF-8-Format. Diese STRING-Formatierung ist üblich bei der MQTT-Kommunikation.

Um auch Sonderzeichen und Texte verschiedenster Sprachen empfangen zu können, wird der Zeichensatz in der Tc3_IotCommunicator-Bibliothek nicht auf den typischen Zeichensatz vom Datentyp STRING beschränkt. Stattdessen wird der Unicode-Zeichensatz als UTF-8-Format in Verbindung mit dem Datentyp STRING verwendet.

Bei Verwendung des ASCII-Zeichensatzes besteht kein Unterschied zwischen der typischen Formatierung in einem STRING und der UTF-8-Formatierung eines STRING.

6.1.1.4 SendDataAsString

Diese Methode wird einmalig aufgerufen, um Daten an den Broker zu senden. Im Gegensatz zu der klassischen SendData [▶ 47]()-Methode wird an dieser Stelle das JSON-Dokument direkt übergeben. So erreicht der Benutzer eine höhere Flexibilität, muss aber im Gegenzug dafür ein korrekt aufgebautes JSON-Dokument an die App schicken.

Die Methode richtet sich an im Umgang mit JSON-Dokumenten erfahrene Nutzer. Bei fehlerhaft formatierten JSON-Dokumenten können die Informationen nicht in der App angezeigt werden.

Syntax

```
METHOD SendDataAsString : BOOL
VAR_INPUT
    sJsonString : POINTER TO STRING;
    nJsonLen    : UINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
SendDataAsString	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
sJsonString	POINTER TO STRING	Pointer zu dem JSON-String, der gesendet werden soll.
nJsonLen	UINT	Länge des JSON-Strings

Mögliche Fehler werden an den Ausgängen bError und hrErrorCode der Bausteininstanz ausgegeben.

Aufbau TwinCAT JSON

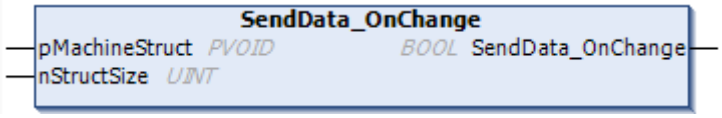
Die TwinCAT IoT Communicator-Produktpalette benutzt für die Kommunikation ein JSON-Format mit dem Namen TwinCAT JSON: Im Folgenden wird der Aufbau eines TwinCAT JSON-Dokuments am Beispiel der Widgets Steckdose [▶ 20] und Lüftung [▶ 25] beschrieben.

```
{
    "Timestamp" : "2022-08-04T07:15:06.176",
    "GroupName" : "Widget Testpage",
    "Values" : {
        "sPageDesc" : "TwinCAT JSON Page",
        "stPlug" : {
            "sDisplayName" : "",
            "bOn" : true,
            "sMode" : "Manual",
            "aModes" : [ "Manual", "Automatic" ]
        },
```

```
        "stVent" : {
             "sDisplayName" : "",
             "bOn" : true,
             "nValue" : 725,
             "nValueRequest" : 400,
             "sMode" : "Manual",
             "aModes" : [ "Manual", "Automatic" ]
        }
    },
    "MetaData" : {
        "sPageDesc" : {
            "iot.DisplayName" : "Info",
            "iot.ReadOnly" : "true"
        },
        "stPlug" : {
            "iot.DisplayName" : "Plug Widget",
            "iot.ReadOnly" : "false",
            "iot.WidgetType" : "Plug",
            "iot.PlugModeVisible" : "true",
            "iot.PlugModeChangeable" : "false"
        },
        "stVent" : {
            "iot.DisplayName" : "Ventilation Widget",
            "iot.ReadOnly" : "false",
            "iot.WidgetType" : "Ventilation",
            "iot.VentilationSliderVisible" : "true",
            "iot.VentilationValueRequestVisible" : "false",
            "iot.VentilationModeVisible" : "true",
            "iot.VentilationModeChangeable" : "false"
         },
        "stVent.nValue" : {
            "iot.Unit" : "ppm",
            "iot.MinValue" : "400",
            "iot.MaxValue" : "1400"
        }
    },
"ForceUpdate":false
}
```

Bereich	Beschreibung
Timestamp	Muss pro Nachricht einen Timestamp im Format: "YYYY-MM-DDThh:mm:ss.fff" enthalten z. B. "2022-08-04T07:15:06.176".
GroupName	Name des Einstiegsknotens des Communicator-Bausteins in der App.
Values	Die anzuzeigenden Werte, angefangen auf der ersten Seite, mit anschließender Verschachtelung.
MetaData	Alles, was in SPS-Attributen umgesetzt ist (bspw. die Konfiguration der Widgets).
ForceUpdate	Optionaler Parameter. Wird beim OnChange-Mechanismus dafür eingesetzt, nach Änderungen ein Update zu triggern. Genauere Informationen unter OnChange-Mechanismen [▶ 41].

6.1.1.5 SendData_OnChange



Diese Methode wird einmalig aufgerufen, um Daten an den Broker zu senden.

Die SendData [▶ 47]-Methode überträgt immer die gesamten Daten als Retain-Nachricht. Mit der SendData_OnChange-Methode hingegen ist es möglich, einzelne Teile der Daten zu übertragen, um Datenverkehr einzusparen. Es ist aber zu beachten, dass die Nachrichten mit der SendData_OnChange-Methode nicht als Retain-Nachricht geschickt werden und somit nur aktuell verbundenen App-Instanzen

bekannt werden. Die Daten müssen im TwinCAT-Projekt bereitgehalten werden und bei Verbindung eines neuen Clients mithilfe der SendData [▶ 47]-Methode als gesamter Datenblock und als Retain-Nachricht geschickt werden.

Es ist an dieser Stelle zu beachten, dass die OnChange-Mechanismen [▶ 41] hauptsächlich für die Verwendung mit den SendDataAsString [▶ 48]-Methoden vorgesehen sind. Mit den SendData-Methoden kann OnChange nur benutzt werden, wenn beim Senden der Struktur eine Struktur verwendet wird, die bis zu den zu ändernden Variablen den gleichen Variablenpfad an den Tag legt.

Syntax

```
METHOD SendData_OnChange : BOOL
VAR_INPUT
    pMachineStruct : PVOID;
    nStructSize    : UINT;
END_VAR
```

Rückgabewert

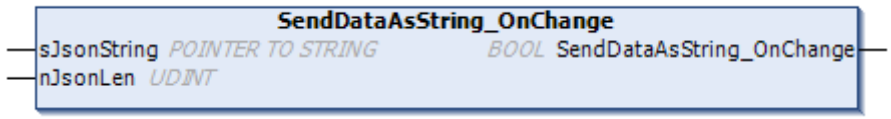
Name	Typ	Beschreibung
SendData_OnChange	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
pMachineStruct	PVOID	Adresse zur Struktur, in der die Variablen des Gerätes deklariert werden.
nStructSize	UINT	Größe der bei pMachineStruct angegebenen Struktur.

Mögliche Fehler werden an den Ausgängen bError und hrErrorCode der Bausteininstanz ausgegeben.

6.1.1.6 SendDataAsString_OnChange



Diese Methode wird einmalig aufgerufen, um Daten an den Broker zu senden. Im Gegensatz zu der klassischen SendData [▶ 47]()-Methode wird an dieser Stelle das JSON-Dokument direkt übergeben. So erreicht der Benutzer eine höhere Flexibilität, muss aber im Gegenzug dafür ein korrekt aufgebautes JSON-Dokument an die App schicken.

Die Methode richtet sich an im Umgang mit JSON-Dokumenten erfahrene Nutzer. Bei fehlerhaft formatierten JSON-Dokumenten können die Informationen nicht in der App angezeigt werden.

Die SendDataAsString [▶ 48]-Methode überträgt die Daten als Retain-Nachricht. Mit der SendDataAsString_OnChange-Methode hingegen ist es möglich, einzelne Teile der Daten zu übertragen, um Datenverkehr einzusparen. Es ist aber zu beachten, dass die Nachrichten mit der SendDataAsString_OnChange-Methode nicht als Retain-Nachricht geschickt werden und somit nur aktuell verbundenen App-Instanzen bekannt werden. Die Daten müssen im TwinCAT-Projekt bereitgehalten werden und bei Verbindung eines neuen Clients mithilfe der SendDataAsString [▶ 48]-Methode als gesamter Datenblock geschickt werden.

Syntax

```
METHOD SendDataAsString_OnChange : BOOL
VAR_INPUT
    sJsonString : POINTER TO STRING;
    nJsonLen    : UINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
SendDataAsString_On Change	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
sJsonString	POINTER TO STRING	Pointer zu dem JSON-String, der gesendet werden soll.
nJsonLen	UINT	Länge des JSON-Strings

Mögliche Fehler werden an den Ausgängen bError und hrErrorCode der Bausteininstanz ausgegeben.

6.1.2 FB_IotCommand

Der Funktionsbaustein stellt Funktionen zur Verfügung, um empfangene Befehle („Commands“) auszuwerten. Er darf nicht instanziiert werden, da er am Ausgang der FB_IotCommunicator [▶ 44]-Instanz bereits deklariert ist und darüber auf die Ausgänge und die empfangenen Befehle zugegriffen wird.

Syntax

Definition:

```
FUNCTION_BLOCK FB_IotCommand
VAR_INPUT
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
    bAvailable  : BOOL // if true, a new command is available
    sVarName    : STRING // Name of variable in currently available command
END_VAR
```

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, sobald eine Fehlersituation eintritt.
hrErrorCode	HRESULT	Liefert bei einem gesetzten bError-Ausgang einen Fehlercode.
bAvailable	BOOL	TRUE, wenn ein neuer Befehl („Command“) verfügbar ist.
sVarName	STRING	Wenn bAvailable TRUE ist, enthält sVarName den Namen der Variablen, die empfangen wurde.

Methoden

Name	Beschreibung
GetValue [▶ 52]	Methode, um an den Wert des Befehls zu kommen, wenn bAvailable TRUE ist
Remove [▶ 52]	Methode, um den aktuell verfügbaren Befehl zu verwerfen

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.0	IPC oder CX (x86, x64, ARM)	Tc3_IotCommunicator

6.1.2.1 GetValue

Diese Methode wird aufgerufen, um an den Wert der Variable im aktuellen Befehl („Command“) zu kommen.

Syntax

```
METHOD GetValue : BOOL
VAR_INPUT
    pValue    : PVOID;
    nSize     : UDINT;
    eDatatype : E_IotCommunicatorDatatype;
END_VAR
```

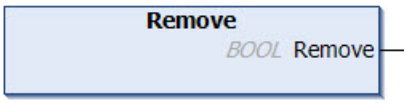
Rückgabewert

Name	Typ	Beschreibung
GetValue	BOOL	Bei erfolgreichem Aufruf liefert die Methode den Rückgabewert TRUE.

Eingänge

Name	Typ	Beschreibung
pValue	PVOID	Adresse zur Variable, in die der empfangene Wert geschrieben werden soll.
nSize	UDINT	Größe der in pValue angegebenen Variablen
eDatatype	E_IotCommunicatorDatatype	Datentyp der in pValue angegebenen Variablen, anhand des Enums E_IotCommunicatorDatatype

6.1.2.2 Remove



Diese Methode wird aufgerufen, um den aktuell verfügbaren Befehl aus dem Speicher zu entfernen.

Rückgabewert

Name	Typ	Beschreibung
Remove	BOOL	Dieser Wert wird bei einem erfolgreichen Aufruf der Methode auf TRUE gesetzt.

6.2 Datentypen

6.2.1 ST_IotCommunicatorTls

TLS-Sicherheitseinstellungen für den MQTT-Client.

Syntax

Definition:

```
TYPE ST_IotCommunicatorTls :
STRUCT
    eVersion           : E_IotCommunicatorTlsVersion := E_IotCommunicatorTlsVersion.tlsv1_2; // TLS
version, which is used
    sCA                : STRING(255); // certificate authority as filename (PEM or DER format) or as
 string (PEM)
    sCert              : STRING(255); // (*optional*) client certificate as filename (PEM or DER for
mat) or as string (PEM)
    sKeyFile           : STRING(255); // (*optional*) client key as filename
    sKeyPwd            : STRING(255);
    bNoServerCertCheck : BOOL; // if FALSE the server certificate is validated (default)
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
eVersion	E_IotCommunicatorTlsVersion	Zu verwendende TLS-Version, anhand des Enums E_IotCommunicatorTlsVersion.
sCA	STRING(255)	Zertifikat der Certificate Authority (CA)
sCert	STRING(255)	Client-Zertifikat, das zur Authentifizierung am Broker verwendet wird (optional)
sKeyFile	STRING(255)	Privater Schlüssel des Clients
sKeyPwd	STRING(255)	Passwort des privaten Schlüssels, falls anwendbar
bNoServerCertCheck	BOOL	Deaktiviert die Überprüfung des Server-Zertifikats auf Gültigkeit. Wenn ohne TLS-Verschlüsselung (HTTP) kommuniziert werden soll, muss dieser Wert auf FALSE bleiben.

6.2.2 E_IotCommunicatorDatatype

Syntax

```
TYPE E_IotCommunicatorDatatype :
{
    type_STRING     :=0,
    type_BOOL       :=1,
    type_SINT       :=2,
    type_INT        :=3,
    type_DINT       :=4,
    type_LINT       :=5,
    type_USINT_BYTE :=6,
    type_UINT_WORD  :=7,
    type_UDINT_DWORD:=8,
    type_ULINT      :=9,
    type_REAL       :=10,
    type_LREAL      :=11
) INT;
END_TYPE
```

Parameter

Name	Typ	Beschreibung
type_STRING	INT	Datentyp STRING.
type_BOOL	INT	Datentyp BOOL.
type_SINT	INT	Datentyp SINT.
type_INT	INT	Datentyp INT.
type_DINT	INT	Datentyp DINT.
type_LINT	INT	Datentyp LINT.
type_USINT_BYTE	INT	Datentyp USINT_BYTE.
type_UINT_WORD	INT	Datentyp UINT_WORD.
type_UDINT_DWORD	INT	Datentyp UDINT_DWORD.
type_ULINT	INT	Datentyp ULINT.
type_REAL	INT	Datentyp REAL.
type_LREAL	INT	Datentyp LREAL.

6.2.3 E_IotCommunicatorTlsVersion

Syntax

```
TYPE E_IotCommunicatorTlsVersion :
{
    tlsv1  :=0,
    tlsv1_1:=1,
    tlsv1_2:=2
) INT;
END_TYPE
```

Parameter

Name	Typ	Beschreibung
tlsv1	INT	TLS Version 1.
tlsv1_1	INT	TLS Version 1.1.
tlsv1_2	INT	TLS Version 1.2.

7 App

Die TwinCAT IoT Communicator App steht im Apple AppStore und Google PlayStore kostenfrei zum Download zur Verfügung.

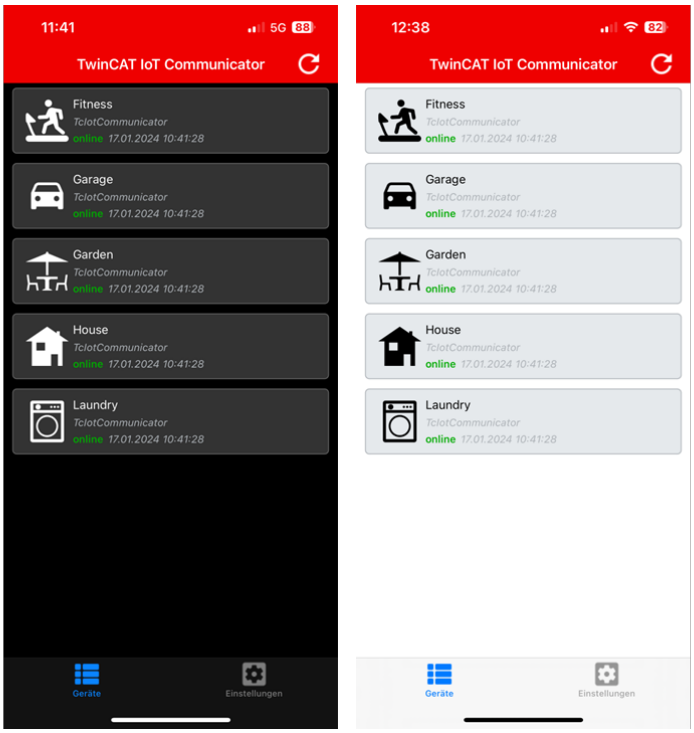
Download on the App Store

GET IT ON Google Play

Google Play and the Google Play logo are trademarks of Google Inc.

Es wird an dieser Stelle empfohlen, bei neuen Projekten sowohl mit der neuesten App-Version als auch mit der neuesten TwinCAT-Version als Engineering zu arbeiten, um auch neue Features nutzen zu können.

Seit der App-Version 1.2.2 wird zusätzlich zum „light mode“ auch der „dark mode“ unterstützt. In welchem der beiden Modi die App dargestellt wird, hängt von den jeweiligen Betriebssystemeinstellungen ab.

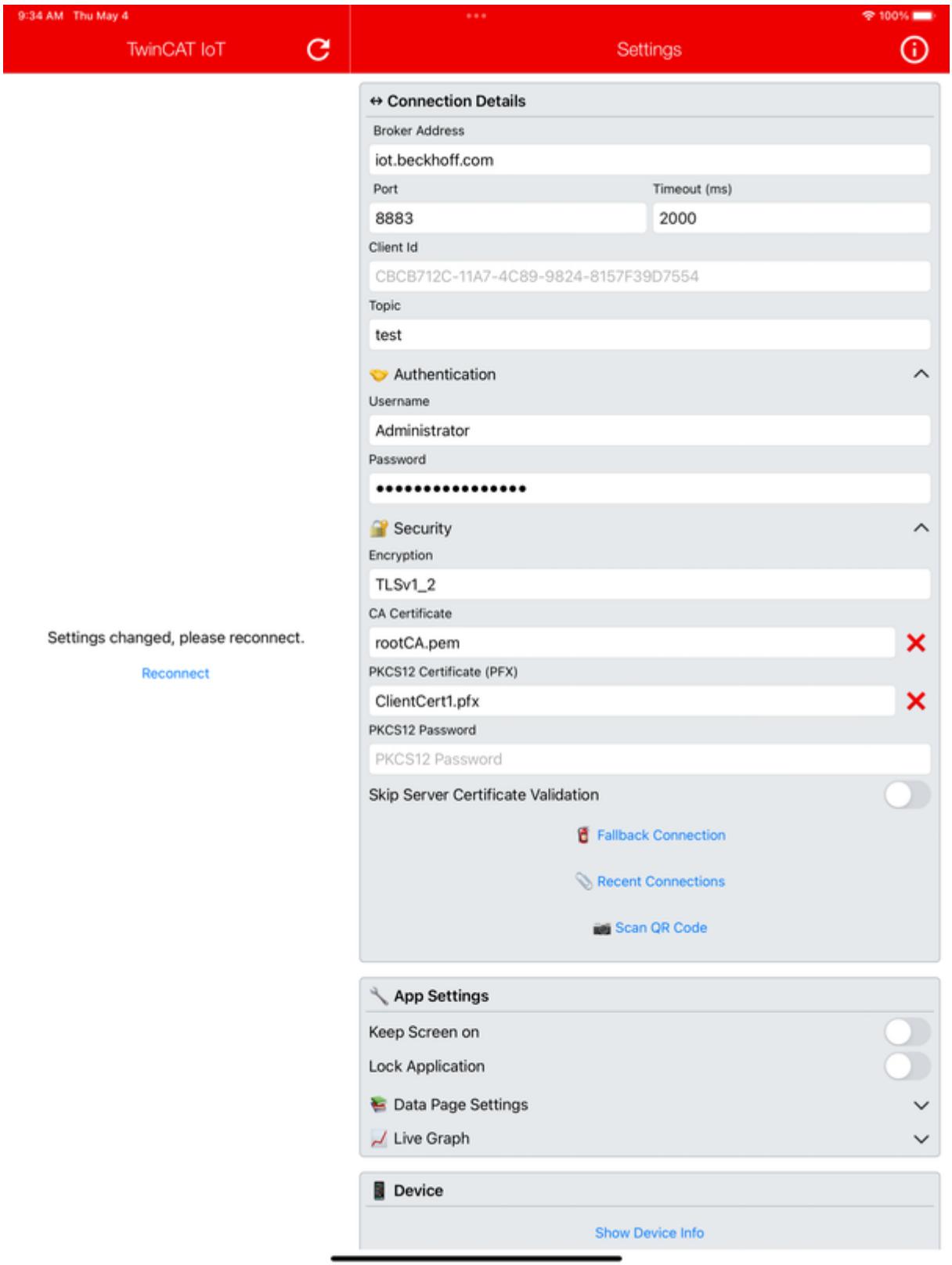


7.1 Einstellungen

Die Einstellungen innerhalb der App teilen sich in drei verschiedene Bereiche auf. Der erste Bereich ermöglicht die Konfiguration der Verbindung zum Broker, im zweiten Bereich lassen sich allgemeine Einstellungen in der App treffen und im dritten und letzten Bereich lassen sich Geräteinfos anzeigen.

7.1.1 Verbindungseinstellungen

Um Daten aus der SPS empfangen zu können, müssen die App und die SPS mit demselben Message Broker verbunden sein. Im Folgenden werden die verschiedenen Einstellungsmöglichkeiten der Verbindung zu diesem Message Broker beschrieben.


9:34 AM Thu May 4
100%
TwinCAT IoT
Settings
Settings changed, please reconnect.
Reconnect
↔ Connection Details
Broker Address
iot.beckhoff.com
Port
Timeout (ms)
8883
2000
Client Id
CBCB712C-11A7-4C89-9824-8157F39D7554
Topic
test
Authentication
Username
Administrator
Password
Security
Encryption
TLSv1_2
CA Certificate
rootCA.pem
PKCS12 Certificate (PFX)
ClientCert1.pfx
PKCS12 Password
PKCS12 Password
Skip Server Certificate Validation
Fallback Connection
Recent Connections
Scan QR Code
App Settings
Keep Screen on
Lock Application
Data Page Settings
Live Graph
Device
Show Device Info

Grundeinstellungen

Einstellung	Bedeutung
Broker Address	IP-Adresse oder Hostname des Message Brokers.
Port	Der Port der Message Brokers. Für gewöhnlich 1883 (MQTT) oder 8883 (MQTT TLS).
Timeout	Diese Einstellung gibt die Zeit an, nachdem die Verbindung zum Message Broker in einen Timeout läuft. Nach dieser Zeit wird bei einer aktiven Fallback Connection die Verbindung zu dem zweiten Message Broker probiert.
Client Id	Die Client-ID der App, mit der die Verbindung zum Message Broker hergestellt wird. Wenn kein benutzerdefinierter Wert eingetragen wird, wird die eindeutige Gerätekennung des mobilen Gerätes verwendet.
Topic	Haupt-Topic, über das die Nachrichten aus dem zugehörigen SPS-Programm kommuniziert werden.

Authentisierung

Je nach Broker-Konfiguration kann es notwendig sein, bei der Verbindungsherstellung Username und Passwort anzugeben. Wenn ein Broker mit der Möglichkeit von anonymen Zugriffen verwendet wird, bleiben diese Felder der Konfiguration leer.

Einstellung	Bedeutung
Username	Benutzername zur Anmeldung am Message Broker
Password	Zugehöriges Passwort für den verwendeten Benutzer

Security

Zusätzlich zur Authentisierung spielt auch die Verschlüsselung von Nachrichten eine wichtige Rolle.

Einstellung	Bedeutung
Encryption	Auswahl des Verschlüsselungsprotokolls.
CA certificate	Referenzierung des CA-Zertifikats als Datei. Unter Android freier Dateizugriff, unter iOS nur in dem Bereich „Auf meinem iPhone“. Weiterführende Informationen unter Installation von CA-Zertifikaten [▶ 59].
PKCS12 Certificate (PFX)	Referenzierung des Client-Zertifikats als Datei. Unter Android freier Dateizugriff, unter iOS nur in dem Bereich „Auf meinem iPhone“. Das Zertifikat muss als PFX-Datei vorliegen, Informationen zur Konvertierung entnehmen Sie gängiger Fachliteratur.
PKCS12 Password	Passwort für die PFX-Datei.
Skip Server Certificate Validation	Diese Einstellung deaktiviert die Validierung des Server-Zertifikats.

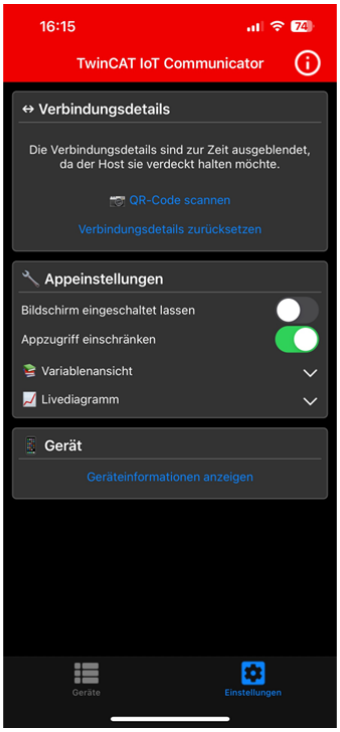
Weitere Einstellungen

Einstellung	Bedeutung
Fallback Connection	Hier kann eine alternative Verbindung zu einem weiteren Message Broker angegeben werden, wenn die primäre Verbindung nicht erreichbar ist. Nach dem weiter oben in den Einstellungen definierten Timeout wird die Verbindung zur Fallback Connection versucht.
Recent Connections	Hier werden die zuletzt konfigurierten Verbindungen angezeigt. Durch Klicken auf die einzelnen Felder werden automatisch die Verbindungsparameter eingefügt. Ein neuer Eintrag wird bei einem neuen Verbindungsversuch hinzugefügt. Wurde mit diesen Parametern schon einmal eine Verbindung hergestellt, wird der Eintrag in der Liste nach oben gesetzt.
Scan QR-Code	Hier kann ein QR-Code mit Verbindungsparametern eingescannt werden. Die Formatierung ist unten in einem gesonderten Abschnitt zu finden.

Verwendung von QR-Codes zur Verbindungsherstellung

Die Einstellungsseite der App bietet eine Möglichkeit, einen QR-Code zu scannen, der die Verbindungsparameter enthält. Die Auswahlmöglichkeiten bezüglich Security sind im Vergleich zu einer händischen Verbindungseinstellung weniger umfangreich. Zusätzlich sollte der Nutzer bedenken, dass je nach Standort jeder über den QR-Code Zugriff erlangen könnte.

Es gibt die Möglichkeit, einen QR-Code mit dem sogenannten Lock-Parameter auszustatten. In diesem Fall kann eine Verbindung über den QR-Code hergestellt werden, dem Nutzer der App werden aber keine Verbindungsdetails angezeigt.



Ein beispielhafter QR-Code sieht wie folgt aus:

http://iotdemo.beckhoff.com/app?&broker=iot.beckhoff.com&port=1883&topic=TOPICNAME

Hier würden die Verbindungsparameter für die Broker-Adresse und den Broker-Port sowie für das Topic „TOPICNAME“ durch das Scannen des QR-Codes eingetragen werden. Die folgende Liste beschreibt die möglichen Parameter, die über die URL abgebildet werden können:

Parameter	Werte
broker	IP-Adresse oder Hostname des Brokers.
port	Port des Message Brokers (im Normalfall 1883 oder 8883).
clientid	Client-ID der App, wenn benötigt.
topic	Topic, auf das die Communicator SPS-Bibliothek veröffentlicht.
user	Benutzername zur Anmeldung am Message Broker.
password	Passwort zur Anmeldung am Message Broker.
lock	Als Werte sind „true“ oder „1“ möglich, wenn das Ausblenden der Verbindungsdetails gewünscht ist.

7.1.1.1 Installation von CA-Zertifikaten

Für das Referenzieren von CA-Zertifikaten unter den beiden unterstützten Betriebssystemen, Android und iOS, müssen diese zunächst installiert werden. Für beide finden Sie im Folgenden eine kurze Anleitung, weitere Informationen entnehmen Sie bitte den Dokumentationen der beiden Hersteller.

Installation von CA-Zertifikaten Android

1. Wählen Sie den Pfad: Einstellungen/Sicherheit/Verschlüsselung und Anmeldedaten/Ein Zertifikat installieren/CA-Zertifikat.
2. Wählen Sie ein CA-Zertifikat aus.
3. Vergeben Sie einen Namen für das Zertifikat (optional).

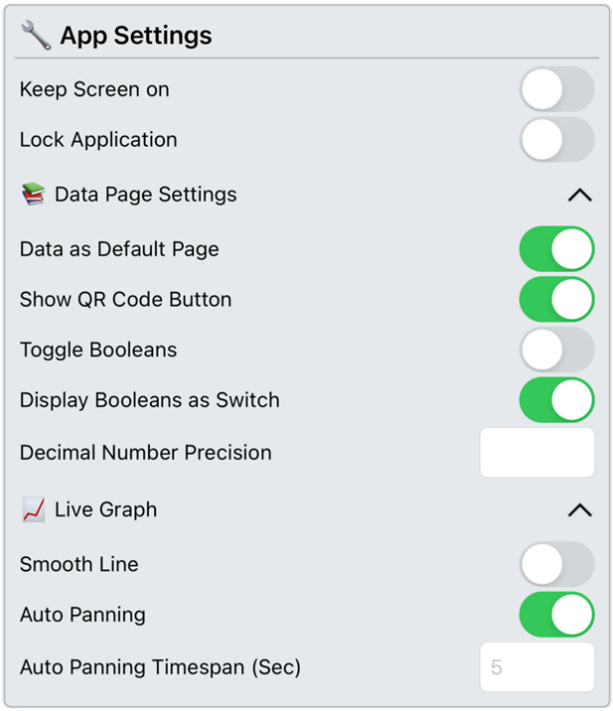
⇨ Das System meldet die erfolgreiche Installation des CA-Zertifikats.

Installation von CA-Zertifikaten iOS

1. Klicken Sie auf die Datei des CA-Zertifikats (muss sich außerhalb des TwinCAT IoT-Ordners befinden). Dadurch wird das Profil geladen.
2. Wählen Sie den Pfad: Einstellungen/Allgemein/Profil.
3. Wählen Sie das Profil des CA-Zertifikats aus.
4. Drücken Sie oben rechts auf **Installieren**.
5. Bestätigen Sie das Installieren durch das Eingeben des Codes.
6. Lesen Sie die Warnhinweise oben rechts und drücken Sie dann auf **Installieren** und bestätigen damit.
7. Wählen Sie den Pfad: Einstellungen/Allgemein/Info/Zertifikatsvertrauenseinstellungen.
8. Aktivieren Sie das installierte Zertifikat unter dem Punkt **Volles Vertrauen für Root-Zertifikate aktivieren**

7.1.2 App-Einstellungen

Neben den Verbindungseinstellungen lassen sich generelle Einstellungen für die App treffen. Diese werden im Folgenden beschrieben.


App Settings
Keep Screen on
Lock Application
Data Page Settings
Data as Default Page
Show QR Code Button
Toggle Booleans
Display Booleans as Switch
Decimal Number Precision
Live Graph
Smooth Line
Auto Panning
Auto Panning Timespan (Sec)
5

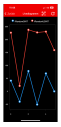
Einstellung	Bedeutung	Default
Keep Screen on	Bei Aktivierung schaltet sich der Bildschirm des mobilen Gerätes nicht aus, solange die App geöffnet ist.	FALSE
Lock Application	Bei Aktivierung wird der eingestellte Sicherungsweg des Betriebssystems verwendet, um die App vor unbefugtem Zugriff zu schützen (Face-ID, Touch-ID, Code, etc.)	FALSE
Data as Default Page	Bei Aktivierung wird beim Öffnen eines Gerätes der Data-Tab geöffnet, sonst wird der Messages-Tab geöffnet.	TRUE
Show QR Code Button	Auf jeder Seite innerhalb eines Devices wird ein Button zum Generieren eines QR-Codes angezeigt.	TRUE
Toggle Booleans	Mit Aktivierung dieser Einstellung können boolsche Variablen direkt durch Drücken auf das Anzeigefeld geschaltet werden. Wenn nicht aktiv, öffnet sich beim Drücken auf die Variable ein Auswahldialog. Diese Einstellung wird nur relevant, wenn “Display Booleans as Switch” auf FALSE ist.	FALSE
Display Booleans as Switch	Boolsche Variablen werden als Schalter angezeigt. Wenn diese Einstellung deaktiviert ist, werden diese Variablen als Textfeld angezeigt.	TRUE
Decimal Number Precision	Beschreibt die Anzahl an Nachkommastellen, auf die REAL- bzw. LREAL-Werte bei der Anzeige in der App gerundet werden. Wenn auf SPS-Seite für eine einzelne Variable ein anderer Wert definiert wird, wird die Decimal Number Precision für diese Variable aus der SPS überschrieben.	Dieser Wert ist standardmäßig nicht gesetzt.
Smooth Line	Bei Anzeige eines Live-Graphen wird bei Aktivierung dieses Features der Verlauf des Graphens abgerundet dargestellt.	FALSE
Auto Panning	Bei Aktivierung wird der Graph auf eine bestimmte Zeitspanne geschnitten.	TRUE
Auto Panning Timespan	Die Zeitspanne auf die ein Graph im Live-View geschnitten wird.	5s

7.2 Geräteübersicht

In der Geräteübersicht werden alle aktuell mit dem gleichen Broker und demselben Topic verbundenen IoT Communicator-Bausteine angezeigt. Durch das Klicken auf eines der Geräte kann dann die Variablenübersicht für dieses Gerät geöffnet werden.

Je nach Aufbau der aus der SPS versendeten Struktur kann es eine oder mehrere Ebenen geben.

Die Anzeige des Graphens wird über das Listensymbol in der oberen rechten Ecke konfiguriert. Nach dem Auswählen der anzuzeigenden Variablen muss anschließend das Play-Symbol gedrückt werden, was das Listensymbol nach Drücken auf dieses ersetzt. Anschließend wird der Graph entsprechend der Einstellungen dargestellt.



8 Beispiele

Die Beispiele teilen sich in zwei Bereiche auf. Zum einen gibt es das Applikationsbeispiel [▶ 63], das ein Beispiel Stück für Stück aufbaut und erklärt. Zum anderen gibt es auf Github zwei Beispiele, die direkt als SPS-Code heruntergeladen und verwendet werden können.

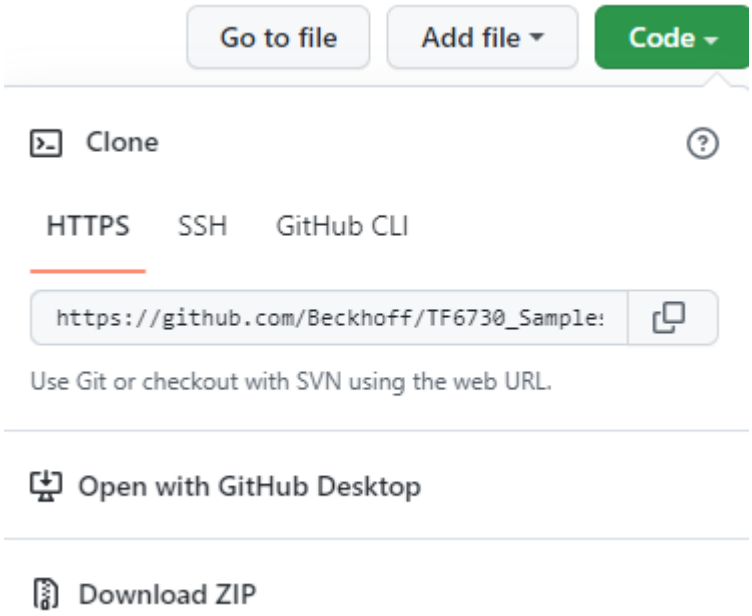
Das erste Sample (TF6730_FullSample) geht dabei auf die Standardfunktionen der App ein, während sich das zweite Sample (TF6730_Widget-Sample) mit den speziellen Erweiterungen für die Gebäudeautomatisierung beschäftigt.

Übersicht

Beispiel	Beschreibung
TF6730_FullSample	Beispiel für die Standard-App-Funktionalitäten.
TF6730_WidgetSample	Beispiel für die Widget-Erweiterungen für die Gebäudeautomatisierung.

Downloads

Der Beispielcode für die in der Übersicht beschriebenen Projekte kann über das entsprechende Repository auf GitHub bezogen werden: https://github.com/Beckhoff/TF6730_Samples. Sie haben dort die Möglichkeit das Repository zu clonen oder ein ZIP-File mit dem Sample herunterzuladen.



8.1 Applikationsbeispiel

Erstellung des SPS-Programms

Struktur definieren

Erzeugen Sie eine Struktur, innerhalb der Sie die zu sendenden Prozessdaten definieren. Weisen Sie dabei den deklarierten Variablen Attribute zu, um deren Darstellung in der App zu definieren (siehe Attribute [▶ 15]).

```
TYPE ST_ProcessData :
STRUCT
    {attribute 'iot.DisplayName' := 'Kitchen Lights'}
    bLamp1 : BOOL;

    {attribute 'iot.DisplayName' := 'Living Room Lights'}
    bLamp2 : BOOL;
```

```
    {attribute 'iot.DisplayName' := 'Outside Temperature'}
    {attribute 'iot.ReadOnly' := 'true'}
    {attribute 'iot.Unit' := 'Celsius'}
    {attribute 'iot.MinValue' := '5'}
    {attribute 'iot.MaxValue' := '30'}
    nTemp  : REAL;
    stSecondLevel : ST_Test;
END_STRUCT
END_TYPE
```

Konfiguration

Deklarieren Sie im Hauptprogramm eine Instanz des Funktionsbausteins FB_IotCommunicator. Definieren Sie die Ausgänge entsprechend Ihren Verbindungsdaten (siehe FB_IotCommunicator [▶ 44]). Deklarieren Sie außerdem die Struktur mit den zu sendenden Prozessdaten sowie eine Instanz des Timer-Funktionsbausteins TON.

```
fbIoT : FB_IotCommunicator := (
    sHostName := 'YOUR_MQTT_BROKER', // MQTT Broker Adress
    nPort := 1883,                   // MQTT Port
    sMainTopic := 'plants',          // Main Topic
    sDeviceName := 'Building 12.3',  // Device Name
    sUser := 'engineer1',            // MQTT Username
    sPassword := 'abcdefg');         // MQTT Password

stData: ST_ProcessData;              // Values to send
timer : TON;                         // Timer to send data
```

Verbindung aufbauen

Rufen Sie im Implementierungsteil des Hauptprogramms über die Instanz des Funktionsbausteins FB_IotCommunicator die Execute-Methode zyklisch auf, um die Verbindung zum Broker aufrechtzuerhalten und somit das Senden und Empfangen von Daten und Nachrichten zu ermöglichen (siehe Execute [▶ 46]).

```
fbIoT.Execute(TRUE);
```

Daten senden

Senden Sie die Prozessdaten mit einer Sample-Rate von 500 ms an den Broker. Rufen Sie dazu die Instanz des Timer-Funktionsbausteins mit den entsprechenden Eingabevariablen sowie die SendData-Methode des Funktionsbausteins FB_IotCommunicator auf (siehe SendData [▶ 47]).

```
timer(IN := NOT timer.Q, PT := T#500MS);

IF fbIoT.bConnected AND timer.Q THEN
    fbIoT.SendData(ADR(stData), SIZEOF(stData));
END_IF
```

Für eine Strukturierung über mehrere Ebenen muss hier eine verschachtelte Struktur übergeben werden. Die folgende Struktur zeigt ein einfaches Beispiel, die Verschachtelung kann je nach Belieben fortgeführt werden.

```
TYPE ST_Test :
STRUCT
    stLevel1 : ST_Level_1;
    nCounter : INT;
END_STRUCT
END_TYPE
```

```
TYPE ST_Level_1 :
STRUCT
    nDoubleCounter: INT;
    stLevel2 : ST_Level_2;
END_STRUCT
END_TYPE
```

Befehle empfangen und auswerten

Rufen Sie den Funktionsbaustein FB_IotCommand und deren Methoden auf, um Befehle zu empfangen und auszuwerten (siehe FB_IotCommand [▶ 51]).

```
IF fbIoT.fbCommand.bAvailable THEN
    IF fbIoT.fbCommand.sVarName = 'bLamp1' THEN
        fbIoT.fbCommand.GetValue(ADR(stData.bLamp1),
SIZEOF(stData.bLamp1), E_IotCommunicatorDatatype.type_BOOL);
    ELSIF fbIoT.fbCommand.sVarName = 'stSecondLevel.nDoubleCounter' THEN
        fbIoT.fbCommand.GetValue(ADR(stData.stSecondLevel. nDoubleCounter),
```

```
SIZEOF(stData.stSecondLevel. nDoubleCounter), E_IotCommunicatorDatatype.type_BOOL);
    END_IF
    fbIoT.fbCommand.Remove();
END_IF
```

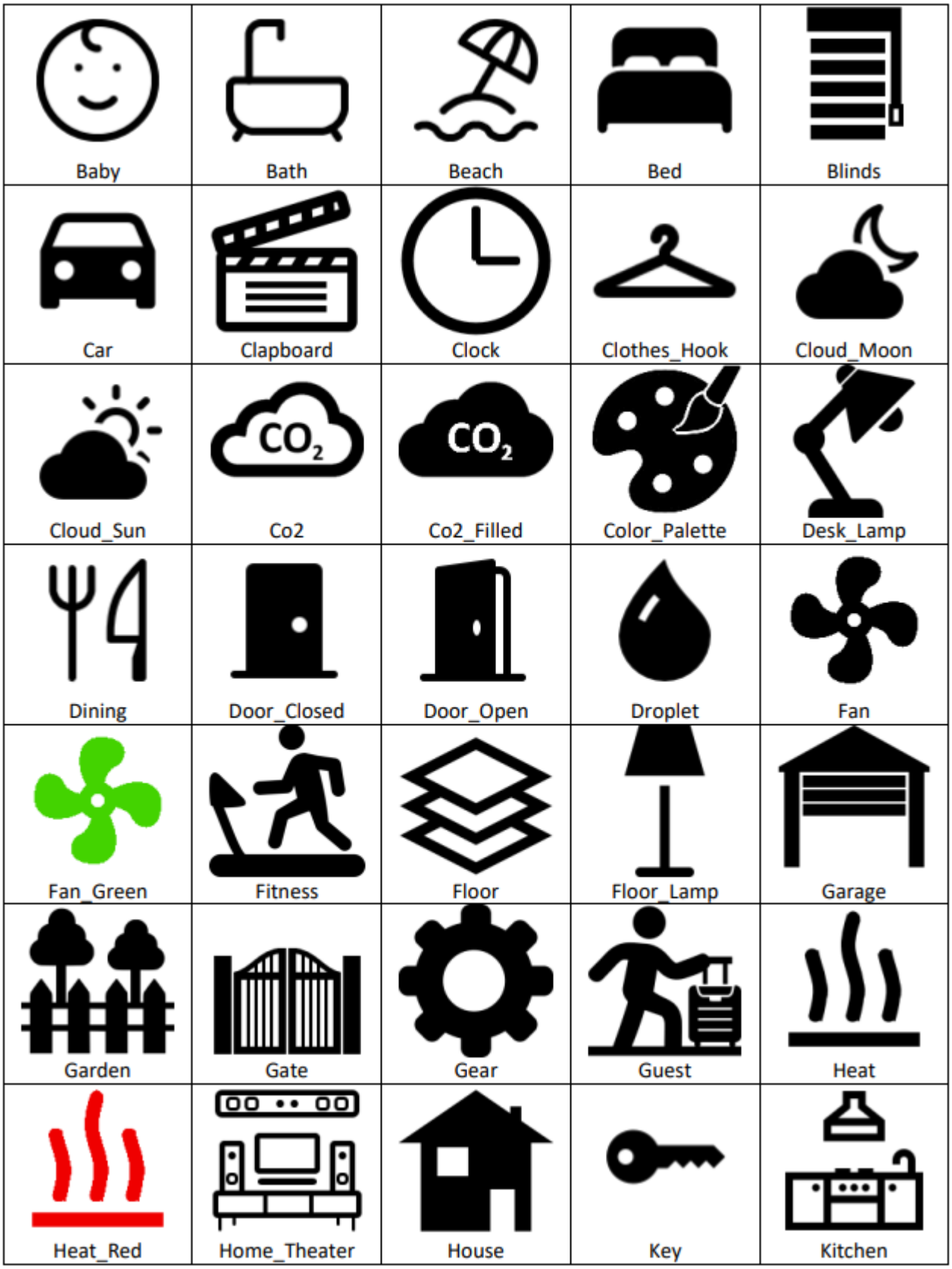
(Push-) Nachrichten senden

Rufen Sie die SendMessage-Methode des Funktionsbausteins FB_IotCommunicator auf, um eine (Push-)Nachricht an den Broker zu senden (siehe SendMessage [▶ 47]).

```
fbIoT.SendMessage('This is a test alarm message!');
```

9 Anhang

9.1 Liste der verfügbaren Icons



9.2 Liste der verfügbaren Farben

Die verfügbaren Farben sind die Farben, die in der Colors Class von Windows dargestellt werden (https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.colors). Dabei können sowohl die Strings als auch die hexadezimalen Werte verwendet werden. Die ersten beiden Stellen der hexadezimalen Darstellung stellen die Deckkraft dar, die weiteren 6 sind die Repräsentation der Farbe.

AliceBlue	#FFF0F8FF	DarkTurquoise	#FF00CED1	LightSeaGreen	#FF20B2AA	PapayaWhip	#FFFFEFD5
AntiqueWhite	#FFFAEBD7	DarkViolet	#FF9400D3	LightSkyBlue	#FF87CEFA	PeachPuff	#FFFFDAB9
Aqua	#FF00FFFF	DeepPink	#FFFF1493	LightSlateGray	#FF778899	Peru	#FFCD853F
Aquamarine	#FF7FFFD4	DeepSkyBlue	#FF00BFFF	LightSteelBlue	#FFB0C4DE	Pink	#FFFFC0CB
Azure	#FFF0FFFF	DimGray	#FF696969	LightYellow	#FFFFFFE0	Plum	#FFDDA0DD
Beige	#FFF5F5DC	DodgerBlue	#FF1E90FF	Lime	#FF00FF00	PowderBlue	#FFB0E0E6
Bisque	#FFFFE4C4	Firebrick	#FFB22222	LimeGreen	#FF32CD32	Purple	#FF800080
Black	#FF000000	FloralWhite	#FFFFFAF0	Linen	#FFFAF0E6	Red	#FFFF0000
BlanchedAlmond	#FFFFEBCD	ForestGreen	#FF228B22	Magenta	#FFFF00FF	RosyBrown	#FFBC8F8F
Blue	#FF0000FF	Fuchsia	#FFFF00FF	Maroon	#FF800000	RoyalBlue	#FF4169E1
BlueViolet	#FF8A2BE2	Gainsboro	#FFDCDCDC	MediumAquamarine	#FF66CDAA	SaddleBrown	#FF8B4513
Brown	#FFA52A2A	GhostWhite	#FFF8F8FF	MediumBlue	#FF0000CD	Salmon	#FFFA8072
BurlyWood	#FFDEB887	Gold	#FFFFD700	MediumOrchid	#FFBA55D3	SandyBrown	#FFF4A460
CadetBlue	#FF5F9EA0	Goldenrod	#FFDAA520	MediumPurple	#FF9370DB	SeaGreen	#FF2E8B57
Chartreuse	#FF7FFF00	Gray	#FF808080	MediumSeaGreen	#FF3CB371	SeaShell	#FFFFF5EE
Chocolate	#FFD2691E	Green	#FF008000	MediumSlateBlue	#FF7B68EE	Sienna	#FFA0522D
Coral	#FFFF7F50	GreenYellow	#FFADFF2F	MediumSpringGreen	#FF00FA9A	Silver	#FFC0C0C0
CornflowerBlue	#FF6495ED	Honeydew	#FFF0FFF0	MediumTurquoise	#FF48D1CC	SkyBlue	#FF87CEEB
Cornsilk	#FFFFF8DC	HotPink	#FFFF69B4	MediumVioletRed	#FFC71585	SlateBlue	#FF6A5ACD
Crimson	#FFDC143C	IndianRed	#FFCD5C5C	MidnightBlue	#FF191970	SlateGray	#FF708090
Cyan	#FF00FFFF	Indigo	#FF4B0082	MintCream	#FFF5FFFA	Snow	#FFFFFAFA
DarkBlue	#FF00008B	Ivory	#FFFFFFF0	MistyRose	#FFFFE4E1	SpringGreen	#FF00FF7F
DarkCyan	#FF008B8B	Khaki	#FFF0E68C	Moccasin	#FFFFE4B5	SteelBlue	#FF4682B4
DarkGoldenrod	#FFB8860B	Lavender	#FFE6E6FA	NavajoWhite	#FFFFDEAD	Tan	#FFD2B48C
DarkGray	#FFA9A9A9	LavenderBlush	#FFFFF0F5	Navy	#FF000080	Teal	#FF008080
DarkGreen	#FF006400	LawnGreen	#FF7CFC00	OldLace	#FFFDF5E6	Thistle	#FFD8BFD8
DarkKhaki	#FFBDB76B	LemonChiffon	#FFFFFACD	Olive	#FF808000	Tomato	#FFFF6347
DarkMagenta	#FF8B008B	LightBlue	#FFADD8E6	OliveDrab	#FF6B8E23	Transparent	#00FFFFFF
DarkOliveGreen	#FF556B2F	LightCoral	#FFF08080	Orange	#FFFFA500	Turquoise	#FF40E0D0
DarkOrange	#FFFF8C00	LightCyan	#FFE0FFFF	OrangeRed	#FFFF4500	Violet	#FFEE82EE
DarkOrchid	#FF9932CC	LightGoldenrodYellow	#FFFAFAD2	Orchid	#FFDA70D6	Wheat	#FFF5DEB3
DarkRed	#FF8B0000	LightGray	#FFD3D3D3	PaleGoldenrod	#FFEEE8AA	White	#FFFFFFFF
DarkSalmon	#FFE9967A	LightGreen	#FF90EE90	PaleGreen	#FF98FB98	WhiteSmoke	#FFF5F5F5
DarkSeaGreen	#FF8FBC8F	LightPink	#FFFFB6C1	PaleTurquoise	#FFAFEEEE	Yellow	#FFFFFF00
DarkSlateBlue	#FF483D8B	LightSalmon	#FFFFA07A	PaleVioletRed	#FFDB7093	YellowGreen	#FF9ACD32
DarkSlateGray	#FF2F4F4F						

9.3 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/tf6730

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com