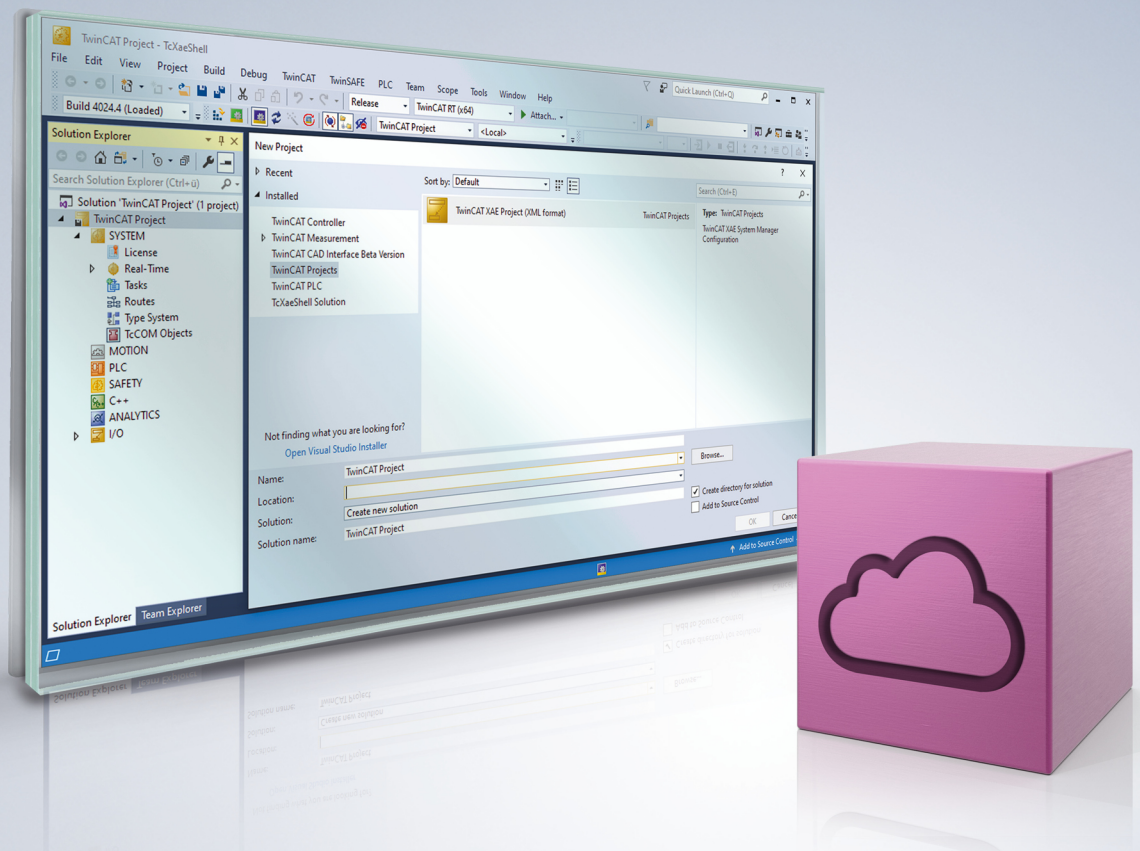
BECKHOFF New Automation Technology

Manual | EN

TF6730-TF6735

TwinCAT 3 | IoT Communicator (-App)



2025-07-08 | Version: 1.10.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

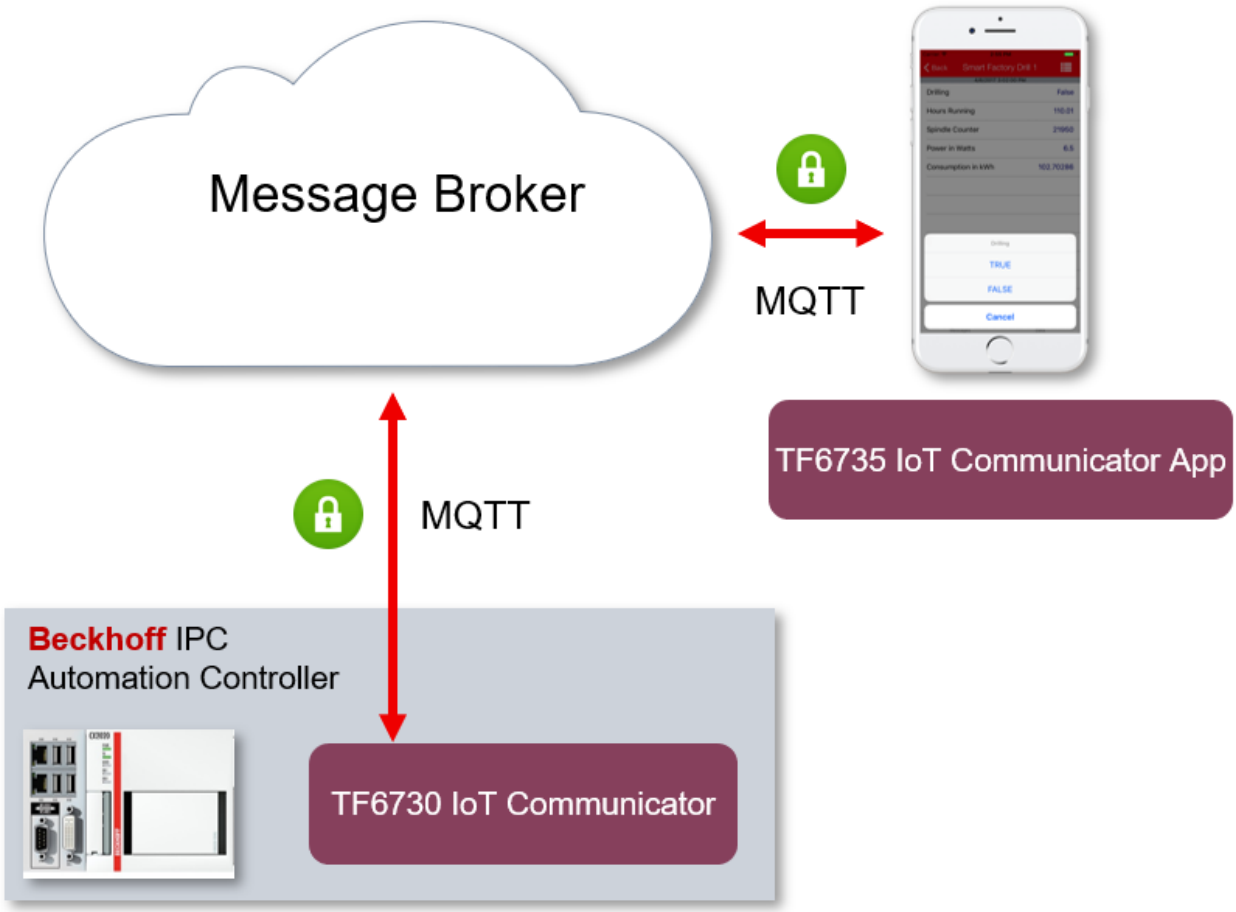
To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

1.4 Documentation issue status

Version	Change
1.10.x	**New:** 4-channel LED [▶ 43]
	New: Favorites [▶ 71]
1.9.x	**New:** Permitted Users [▶ 15]
	New: Energy monitoring [▶ 41]
	New: Use of UTF-8 characters [▶ 50]
	New: User authorizations [▶ 51]
1.8.x	**New:** RGBW lighting [▶ 34]
	New: Bar chart [▶ 36]
	New: OnChange mechanisms [▶ 48]
	New: Arrays [▶ 16]
	New: SendData_OnChange [▶ 58] , SendDataAsString_OnChange [▶ 59]

2 Overview

The function blocks of the Tc3_IoTCommunicator PLC library can be used to realize data exchange between the local TwinCAT PLC and a mobile end device (smart device) via an MQTT message broker. Symbols can be sent and received. Messages can be stored on the broker and read or deleted via the smart device. To this end, the TwinCAT IoT Communicator app must be installed and running on the mobile end device.



The TwinCAT IoT Communicator app can be downloaded free of charge from the Apple AppStore or Google PlayStore.

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 System requirements

Technical data	Description
Operating system	Windows 7/10, Windows Embedded Standard 7, Windows CE 7
Target platform	PC architecture (x86, x64 or ARM)
TwinCAT version	TwinCAT 3.1 build 4022.0 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	TF6730 TC3 IoT Communicator
TwinCAT library to be integrated	TC3_IotCommunicator

3.2 Installation

TwinCAT 3.1 Build 4022 and 4024

All the required components are supplied directly with the TwinCAT Setup.

- TwinCAT XAE Setup: Contains the MQTT driver and the PLC library (Tc3_IotCommunicator).
- TwinCAT XAR Setup: Contains only the MQTT driver.

TwinCAT 3.1 Build 4026

The MQTT driver is already included in the TwinCAT Standard Workload. The PLC library Tc3_IotCommunicator can be installed via the package TwinCAT.XAE.PLC.Lib.Tc3_IotCommunicator.

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, you can install this function via the TwinCAT Package Manager, see Installation documentation.

Normally you install the function via the corresponding workload; however, you can also install the packages contained in the workload individually. This documentation briefly describes the installation process via the workload.

Command line program TcPkg

You can use the TcPkg **C**ommand **L**ine **I**nterface (CLI) to display the available workloads on the system:

```
tcpkg list -t workload
```

You can use the following command to install the Workload of the TF6730 IoT Communicator function.

```
tcpkg install TF6730.IotCommunicator.XAE
```

TwinCAT Package Manager UI

You can use the **U**ser **I**nterface (UI) to display all available workloads and install them if required.
To do this, follow the corresponding instructions in the interface.

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

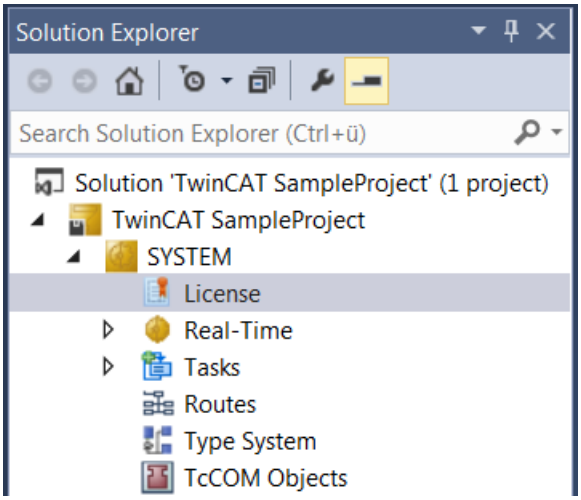
Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

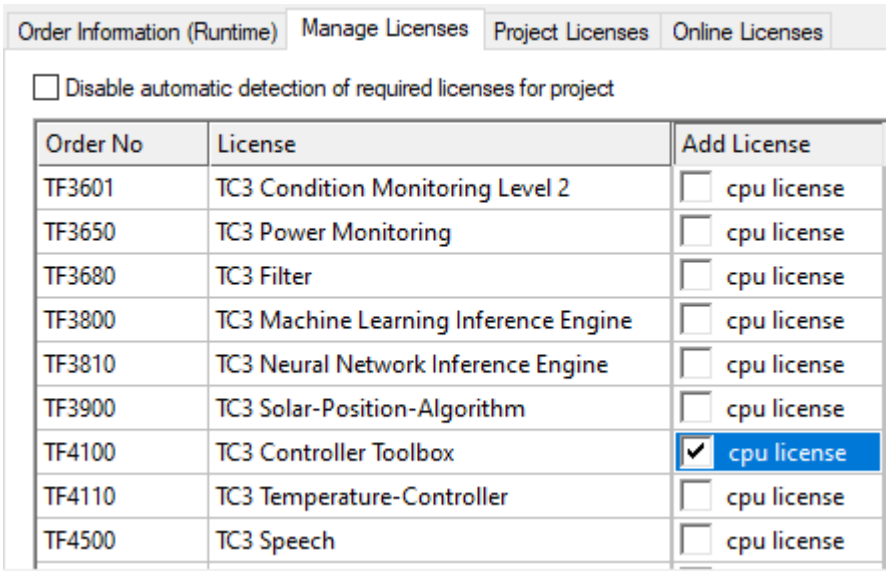
Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



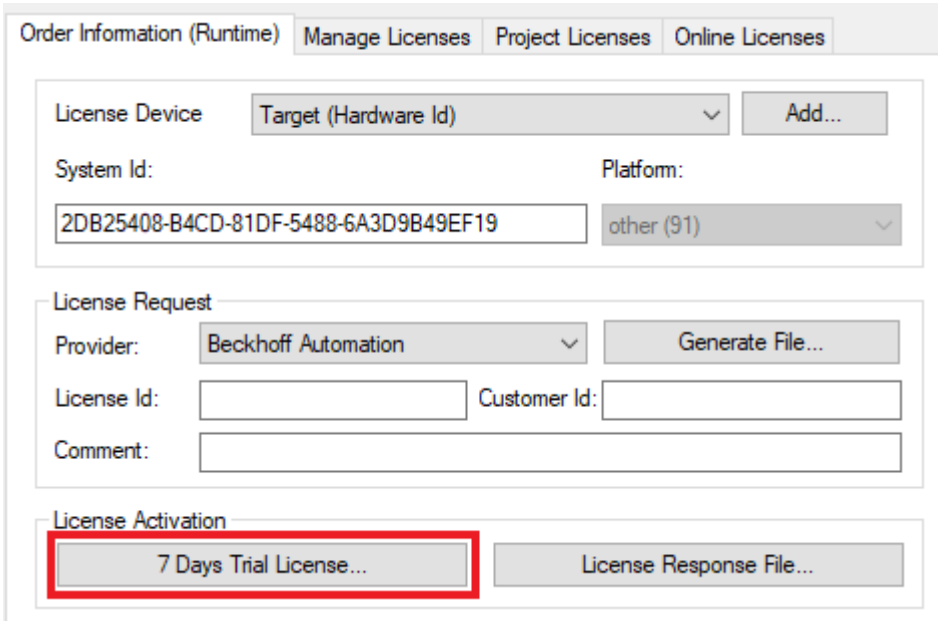
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



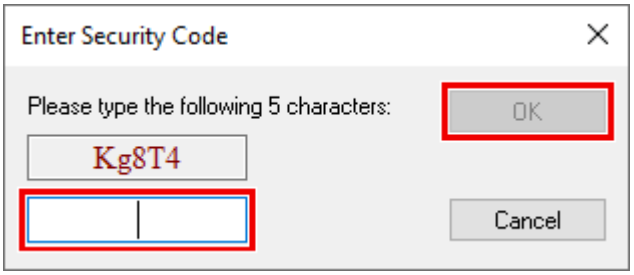
Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 Technical introduction

4.1 MQTT

4.1.1 Broker

An MQTT broker is required to exchange or synchronize process data and messages with a smart device.

The MQTT broker must be accessible via the IP address or host name from the TwinCAT PLC and the mobile device. TwinCAT and smartphone do not have to be connected directly.

MQTT is a publisher/subscriber-based communication protocol, which enables message-based transfer between applications. The message broker is a central component of this transfer type, which distributes messages between the individual applications or the sender and receiver of a message. The message broker decouples the sender and receiver, so that it is not necessary for the sender and receiver to know their respective address information. During sending and receiving all communication devices contact the message broker, which handles the distribution of the messages.

MQTT broker requirements for TC3 IoT Communicator

For optimal use of the TwinCAT IoT Communicator app, the MQTT broker should meet the following requirements:

- MQTT protocol version 3.1.1 (see OASIS standard specification)
- Clients require access to the topic (see Topic structure)
- Retain messages and Quality of Service 0 & 1 (see Quality of Service [▶ 12])

4.1.2 Quality of Service

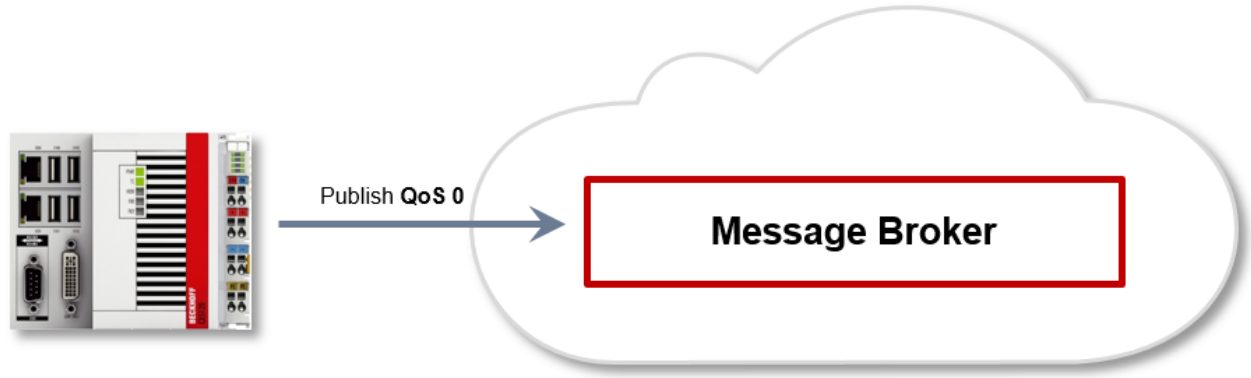
Quality of Service (QoS) is an arrangement between the sender and receiver of a message with regard to guaranteeing of the message transfer. MQTT features three different levels:

- 0 – not more than once
- 1 – at least once
- 2 – exactly once

Both types of communication (publish/subscribe) with the message broker must be taken into account and considered separately. The QoS level that a client uses for publishing a message is set by the respective client. When the broker forwards the message to client that has subscribed to the topic, the subscriber uses the QoS level that was specified when the subscription was established. This means that a QoS level that may have been specified as 2 by the publisher can be “overwritten” with 0 by the subscriber.

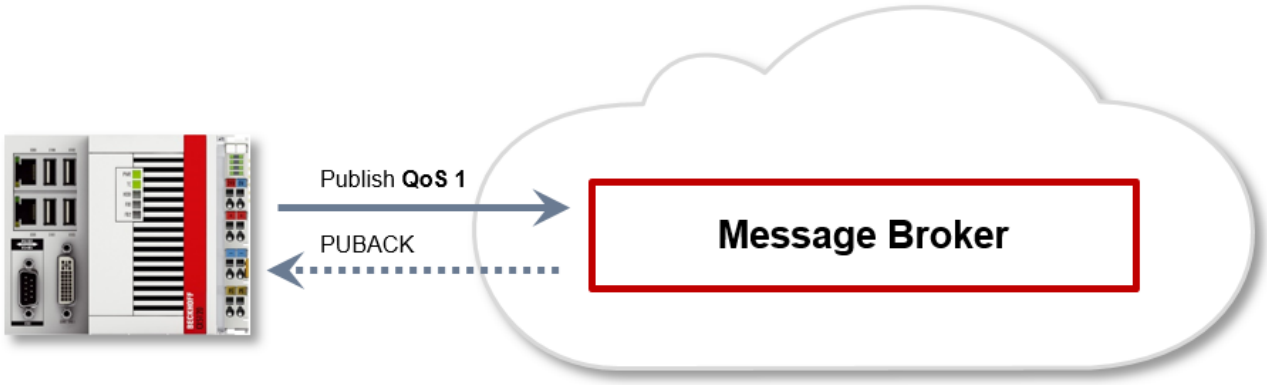
QoS Level 0

At this QoS level the receiver does not acknowledge receipt. The message is not sent a second time.



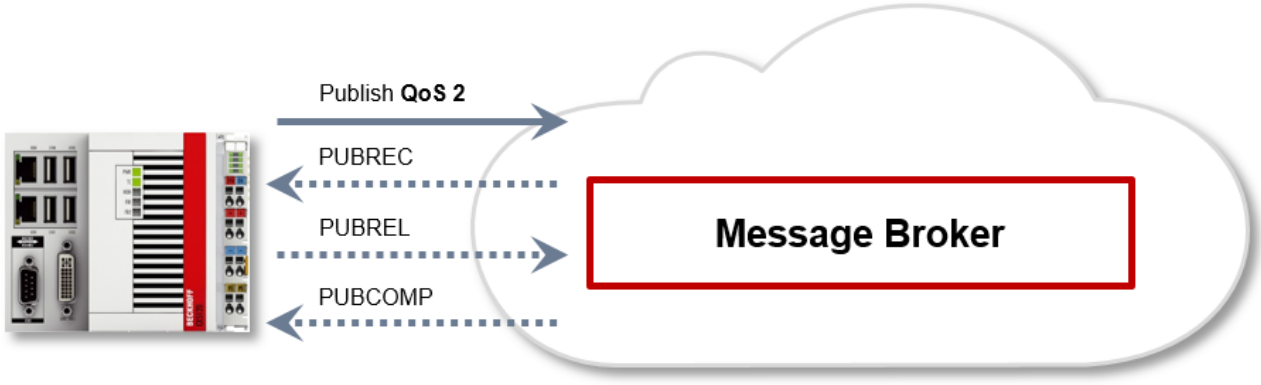
QoS Level 1

At this QoS level the system guarantees that the message arrives at the receiver at least once, although the message may arrive more than once. The sender stores the message internally until it has received an acknowledgment from the receiver in the form of a PUBACK message. If the PUBACK message fails to arrive within a certain time, the message is resent.



QoS Level 2

At this QoS level the system guarantees that the message arrives at the receiver no more than once. On the MQTT side this is realized through a handshake mechanism. QoS level 2 is the safest level (from a message transfer perspective), but also the slowest. When a receiver receives a message with QoS level 2, it acknowledges the message with a PUBREC. The sender of the message remembers it internally until it has received a PUBCOMP. This additional handshake (compared with QoS 1) is important for avoiding duplicate transfer of the message. Once the sender of the message receives a PUBREC, it can discard the initial publish information, since it knows that the message was received once by the receiver. In other words, it remembers the PUBREC internally and sends a PUBREL. Once the receiver has received a PUBREL, it can discard the previously remembered states and respond with a PUBCOMP, and vice versa. Whenever a package is lost, the respective communication device is responsible for resending the last message after a certain time.



4.2 Security

The MQTT specification offers MQTT clients the option to use user name/password authentication with the message broker. Common cryptography mechanisms such as TLS (Transport Layer Security) can be used to provide additional protection for the data communication between client and message broker.

4.2.1 Authentication

The TC3_IoTCommunicator PLC library and the TwinCAT IoT Communicator app can use an authentication mechanism, which is standardized and implemented in the MQTT protocol (see OASIS standard specification). The PLC library and the app use MQTT protocol version 3.1.1.

NOTICE

Authentication does not guarantee protection against cyber attacks

In addition to authentication, TLS encryption should be implemented. Otherwise, the user name and password are transmitted in plain text. (See Encryption [▶ 14])

4.2.2 Encryption

Encryption and authentication via TLS can be accomplished through a certificate authority (CA). The CA provides a signature via the public key for the message broker (the so-called server key) and usually also for all connecting clients. All communication devices can then trust each other, because the issuing certificate authority is trusted. Depending on the message broker, an MQTT client may connect without a dedicated client certificate. In this case the client uses the public key of the issuing certificate authority when it establishes a connection to the broker.

5 Configuration

5.1 Attributes

The attributes listed below are the generally applicable attributes. The attributes introduced specifically for the widgets are described at Widgets [▶ 16].

Display name of the variable (iot.DisplayName)

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

Defines the name to be displayed in the app for this variable. If this attribute is not specified, the PLC variable name is displayed in the app.

Unity of variable (iot.Unit)

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

Defines the unit behind the value of the variable in the app. If this attribute is not specified, the unit behind the value remains empty.

Variable cannot be changed (iot.Readonly)

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

Defines whether the variable can be changed from the app. If this attribute is specified with the value TRUE, the variable can no longer be changed, and a padlock symbol appears next to the variable name. If this attribute is not specified, the variable can be changed by default.

User configuration (IoT.PermittedUsers)

Syntax: {attribute 'iot.PermittedUsers' := 'User1,User2'}

By default, every variable (regardless of whether it is a structure or simple data type) can be seen by every user. If this attribute is added, only the specified users can see the variable in their app.

Icon of a nested structure (iot.NestedStructIcon)

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

Defines the icon for the start page of a nested structure. By default, the TwinCAT CD is displayed. The available icons are listed in the List of available icons [▶ 80].

Minimum and maximum value of the variable (iot.MinValue and iot.MaxValue)

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

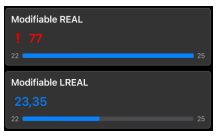
Defines a minimum and maximum value for numerical variables. If both attributes ('MinValue' AND 'MaxValue') are specified, a progress bar in the app shows the progress of the current value with respect to the minimum and maximum value.

NOTICE

Progress indicator

The minimum and maximum value define the range covered by the progress bar in the app. The value can be higher or lower than the values specified in the PLC.

When a value goes leaves its prescribed range, it is highlighted in the app with a value in red. In the following screenshot a value has left its defined range.

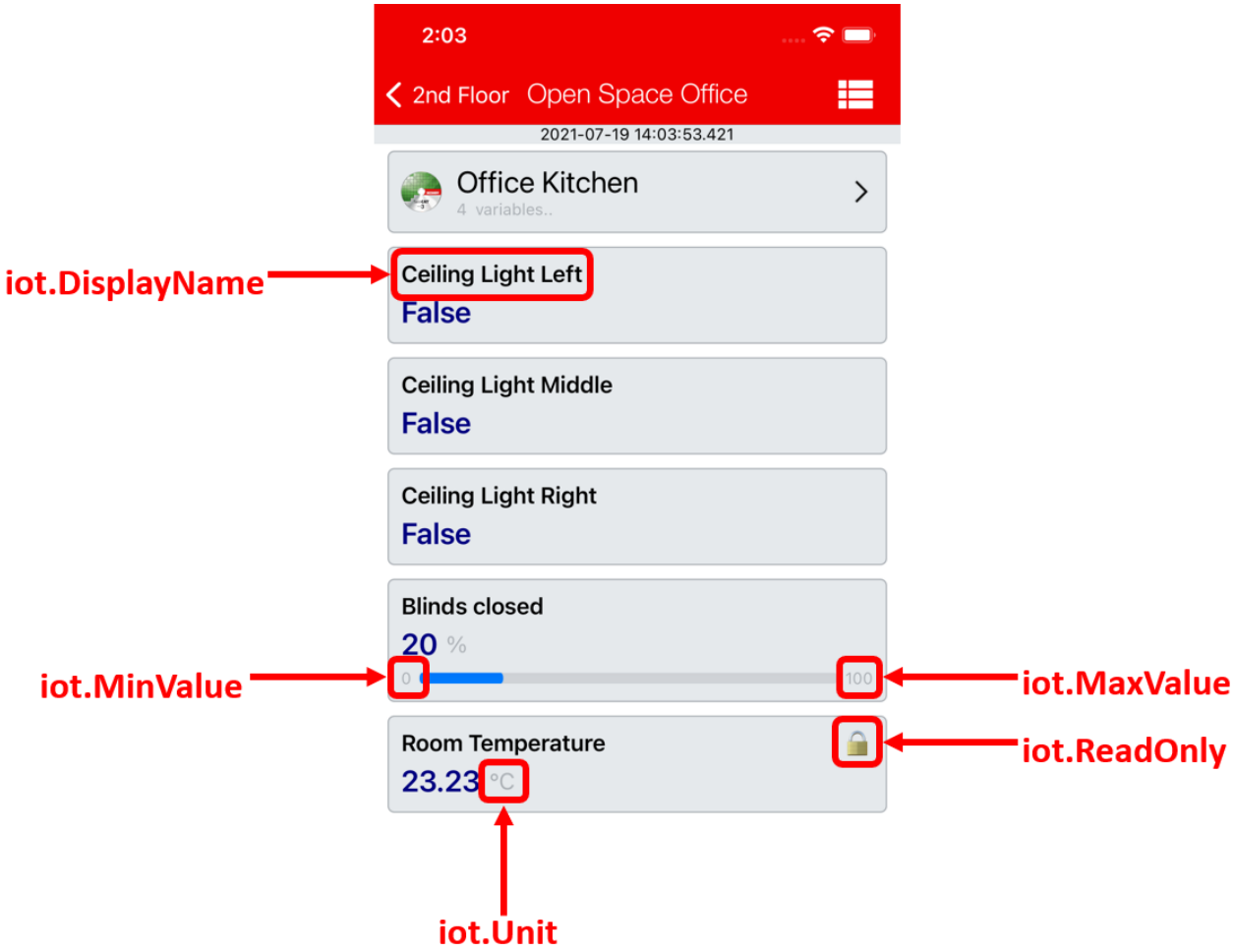


Limitation of the decimal places at a variable (iot.DecimalPrecision)

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

Defines a number of decimal places to which a floating-point number is rounded. This setting overwrites any existing app setting for the respective variable.

Example



See also: examples > Application sample [▶ 77]

5.2 Arrays

One-dimensional arrays of simple data types are supported. If one of these arrays is too long to be displayed in a field in the Communicator app, the list of array values can be opened in a pop-up window by clicking on the field.

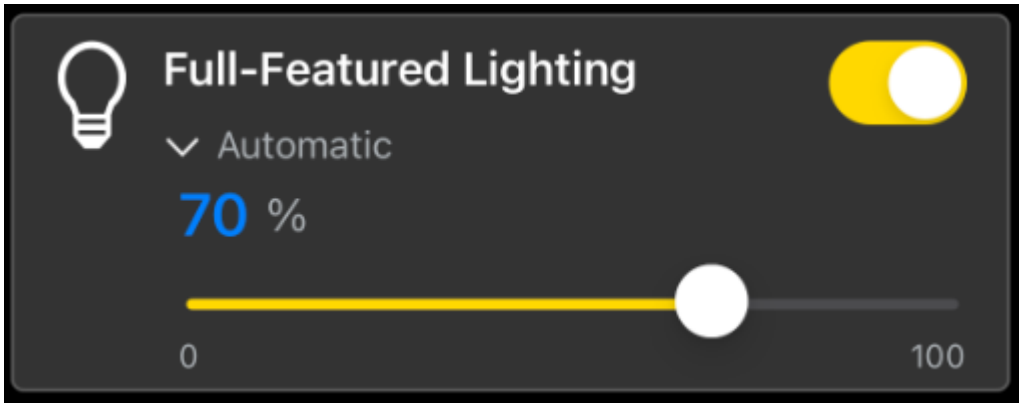
Arrays of structures and multi-dimensional arrays of any data type are currently not supported.

5.3 Widgets

The widgets add functions to the app for operating a building automation system. The widgets are first available with app version 1.4.0 and Tc3_IotCommunicator library version 1.1.14.0 and will be extended with more widgets in following versions.

5.3.1 Lighting

The described widget is suitable for displaying light sources in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

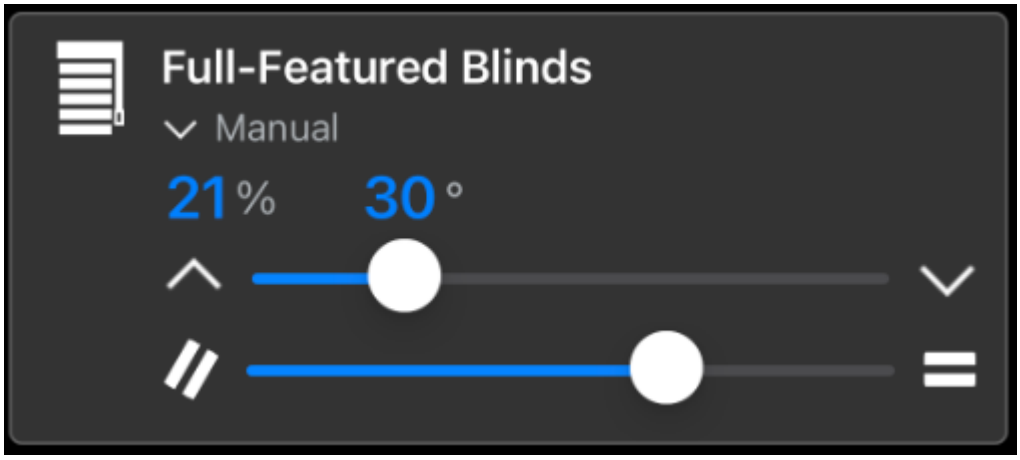
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by *sDisplayName* as soon as *sDisplayName* is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: *Lighting*.
iot.LightValueVisible	BOOL	Determines whether the dimming value is displayed (TRUE) or not (FALSE).
iot.LightSliderVisible	BOOL	Determines whether the slider is displayed (TRUE) or not (FALSE).
iot.LightModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.LightModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bLight	BOOL	Switches the lighting on (TRUE) or off (FALSE).	Toggle switch top right.
iot.Unit	STRING	Unit of the dimming value.	Unit after the numerical value.
iot.MinValue	INT	Lower limit of the dimming value.	On the left side under the slider.
iot.MaxValue	INT	Upper limit of the dimming value.	On the right side under the slider.
nLight	INT	Dimming value of the widget.	Display in the numerical value and additionally display in the filling of the slider.
sMode	STRING	Mode of lighting.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.2 Blinds

The described widget is suitable for displaying blinds in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or additionally also write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: Blinds.
iot.BlindsPositionValueVisible	BOOL	Determines whether the position value is displayed (TRUE) or not (FALSE).
iot.BlindsPositionSliderVisible	BOOL	Determines whether the slider for the position value is displayed (TRUE) or not (FALSE).
iot.BlindsAngleValueVisible	BOOL	Determines whether the angle value is displayed (TRUE) or not (FALSE).
iot.BlindsAngleSliderVisible	BOOL	Determines whether the slider for the angle value is displayed (TRUE) or not (FALSE).
iot.BlindsModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.BlindsModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bActive	BOOL	Intended to detect activation of the blinds.	Click on the area of the widget where there are no other control elements.
bPositionUp	BOOL	Intended for raising the blinds.	Button on the left side of the top slider.
bPositionDown	BOOL	Intended for lowering the blinds.	Button on the right side of the top slider.
bAngleUp	BOOL	Intended for setting the angle value in the direction of the minimum value.	Button on the left side of the bottom slider.
bAngleDown	BOOL	Intended for setting the angle value in the direction of the maximum value.	Button on the right side of the bottom slider.
iot.Unit	STRING	Unit of the position value.	Unit after the first numerical value.
iot.MinValue	INT	Lower limit of the position value.	Only shown in the PLC.
iot.MaxValue	INT	Upper limit of the position value.	Only shown in the PLC.
nPositionValue	INT	Position value of the blinds.	The first of the two numerical values.
nPositionRequest	INT	Target value of the position of the blinds.	The value sent to the app at the moment of releasing the top slider.
iot.Unit	STRING	Unit of the angle value.	Unit after the second numerical value.
iot.MinValue	INT	Lower limit of the angle value.	Only shown in the PLC.
iot.MaxValue	INT	Upper limit of the angle value.	Only shown in the PLC.
nAngleValue	INT	Angle value of the blinds.	The second of the two numerical values.
nAngleRequest	INT	Target value of the position of the blinds.	The value sent to the app at the moment of releasing the bottom slider.
sMode	STRING	Mode of the blinds.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.3 Blinds (simplified)

The described widget is suitable for displaying blinds in the app. The various configuration options are described below. In the figure all available features of the widget are active.

Compared to the other version of the blinds widget, this version is simplified. It offers fewer options, but it has a simpler display.

The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'SimpleBlinds'}
{attribute 'iot.BlindsPositionSliderVisible' := 'true'}
stSimpleBlindsWidgetSample : ST_SimpleBlindsWidgetSample;
```

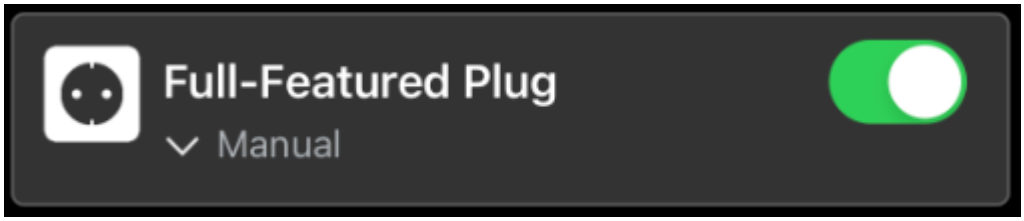
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or additionally also write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: SimpleBlinds.
iot.BlindsPositionSliderVisible	BOOL	Determines whether the slider for the position value is displayed (TRUE) or not (FALSE).

```
TYPE ST_SimpleBlindsWidgetSample :
STRUCT
    sDisplayName     : STRING := '';
    bPositionUp      : BOOL;
    bPositionDown    : BOOL;
    {attribute 'iot.MinValue' := '0'}
    {attribute 'iot.MaxValue' := '100'}
    nPositionValue   : INT;
END_STRUCT
END_TYPE
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Specifies the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bPositionUp	BOOL	Intended for raising the blinds.	Button labeled "Up".
bPositionDown	BOOL	Intended for lowering the blinds.	Button labeled "Down".
iot.MinValue	INT	Lower limit of the position value.	This value is only required in the PLC so that the status display can be scaled correctly.
iot.MaxValue	INT	Upper limit of the position value.	This value is only required in the PLC so that the status display can be scaled correctly.
nPositionValue	INT	Position value of the blinds.	This value is only available in the PLC and is required to display the status below the buttons.

5.3.4 Socket

The described widget is suitable for displaying a socket in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

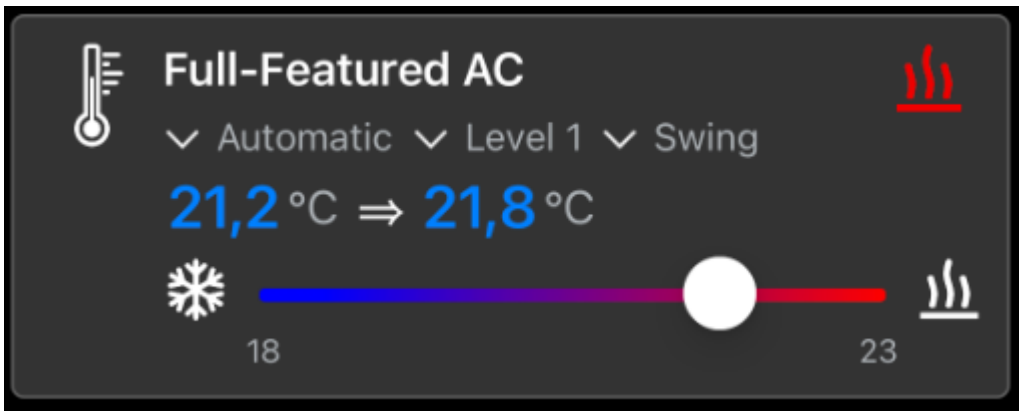
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by *sDisplayName* as soon as *sDisplayName* is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: *Plug*.
iot.PlugModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.PlugModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (FALSE).

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

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bOn	BOOL	Switches the socket on (TRUE) or off (FALSE).	Toggle switch top right.
sMode	STRING	Mode of the socket.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.5 Air conditioning system

The described widget is suitable for displaying air conditioning systems in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

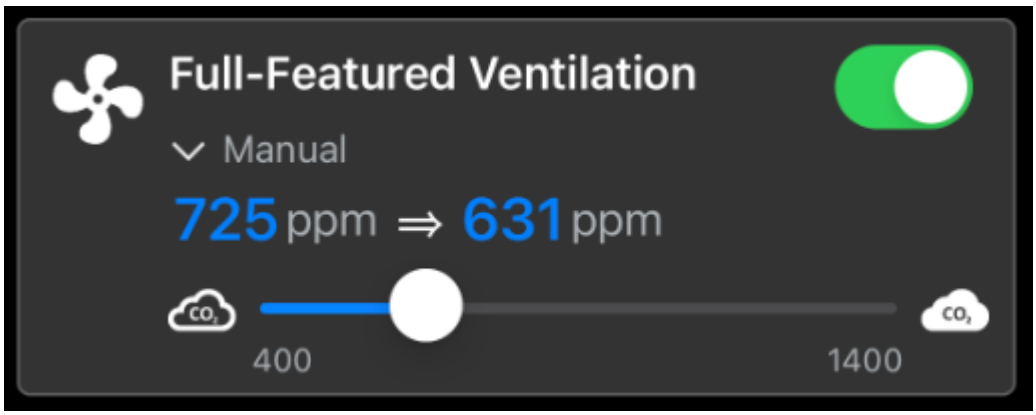
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: *AC*.
iot.ACValueRequestVisible	BOOL	Determines whether the target value is displayed behind the current temperature value (TRUE) or not (FALSE).
iot.ACSliderVisible	BOOL	Determines whether the slider is displayed (TRUE) or not (FALSE).
iot.ACModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.ACModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (FALSE).
iot.ACModeStrengthVisible	BOOL	Determines whether the mode for the strength is displayed (TRUE) or not (FALSE).
iot.ACModeStrengthChangeable	BOOL	Determines whether the mode for the strength is adjustable (TRUE) or not (FALSE).
iot.ACModeLamellaVisible	BOOL	Determines whether the mode for the slats is displayed (TRUE) or not (FALSE).
iot.ACModeLamellaChangeable	BOOL	Determines whether the mode for the slats is adjustable (TRUE) or not (FALSE).
iot.DecimalPrecision	INT	Sets the number of decimal places. This setting overwrites the setting at the 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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
nAcMode	INT	Determines the mode of the AC widget.	Icon top right. 0: no icon 1: Cooling (blue) 2: Ventilating (green) 3: Heating (red) 4: Cooling off (gray) 5: Ventilating off (gray) 6: Heating off (gray)
iot.Unit	STRING	Unit of the temperature value.	Unit behind both numerical values.
iot.MinValue	INT	Lower limit of the temperature range.	On the left side under the slider.
iot.MaxValue	INT	Upper limit of the temperature range.	On the right side under the slider.
iot.DecimalPrecision	INT	Number of decimal places for the temperature values. Overwritten by the DecimalPrecision on the widget and applies to both temperature values.	For both temperature values.
nTemperature	LREAL	Current temperature value.	The number on the left side of the arrow.
nTemperatureRequest	LREAL	Requested temperature value, possible via the slider in steps of 0.1.	The temperature requested via the slider is then displayed on the right side of the arrow.
sMode	STRING	Mode of the air conditioning system.	The currently displayed mode (left).
aModes	ARRAY [0..n] OF STRING	Array of the different general modes that can be set by the user.	By pressing on the current mode (left), the adjustable modes can be displayed.
sMode_Strength	STRING	Mode of the stage of air conditioning system.	The currently displayed mode (centered).
aModes_Strength	ARRAY [0..n] OF STRING	Array of the different step modes adjustable by the user.	By pressing on the current mode (centered), the adjustable modes can be displayed.
sMode_Lamella	STRING	Mode of the louvers of the air conditioning system.	The currently displayed mode (right).
aModes_Lamella	ARRAY [0..n] OF STRING	Array of the different louver modes adjustable by the user.	By pressing on the current mode (right), the adjustable modes can be displayed.

5.3.6 Ventilation

The described widget is suitable for displaying a ventilation in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

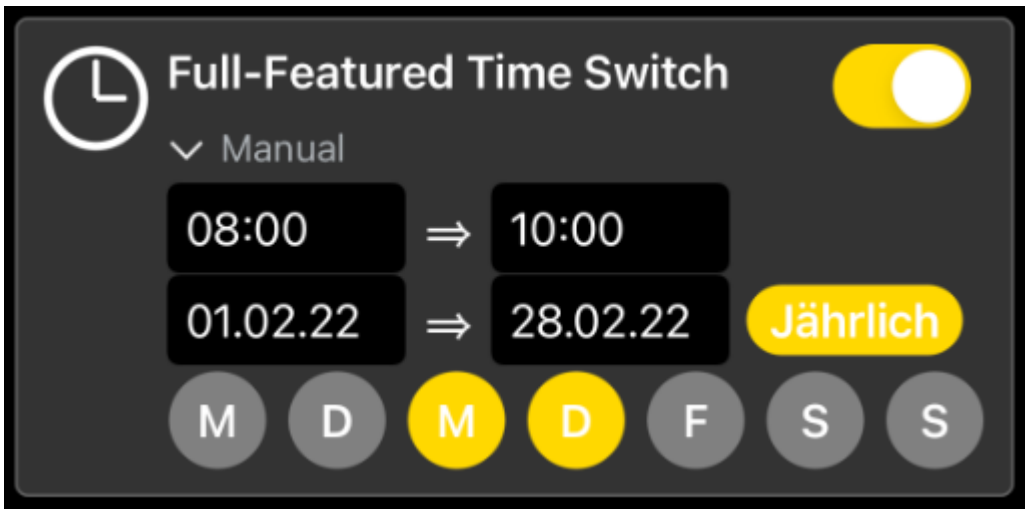
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: Ventilation.
iot.VentilationValueRequestVisible	BOOL	Determines whether the requested air value (CO2 concentration) is displayed (TRUE) or not (FALSE).
iot.VentilationSliderVisible	BOOL	Determines whether the slider is displayed (TRUE) or not (FALSE).
iot.VentilationModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.VentilationModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bOn	BOOL	Switches the ventilation on (TRUE) or off (FALSE).	Toggle switch top right.
iot.Unit	STRING	Unit of air value (of CO2 concentration).	Unit behind both numerical values.
iot.MinValue	INT	Lower limit of the air value (the CO2 concentration).	On the left side under the slider.
iot.MaxValue	INT	Upper limit of the air value (of the CO2 concentration).	On the right side under the slider.
nValue	INT	Current air value (CO2 concentration).	The number to the left of the arrow.
nValueRequest	INT	Requested air value (CO2 concentration) via the slider.	The number requested via the slider is subsequently displayed on the right side of the arrow.
sMode	STRING	Mode of ventilation.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.7 Timer

The described widget is suitable for displaying a timer in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

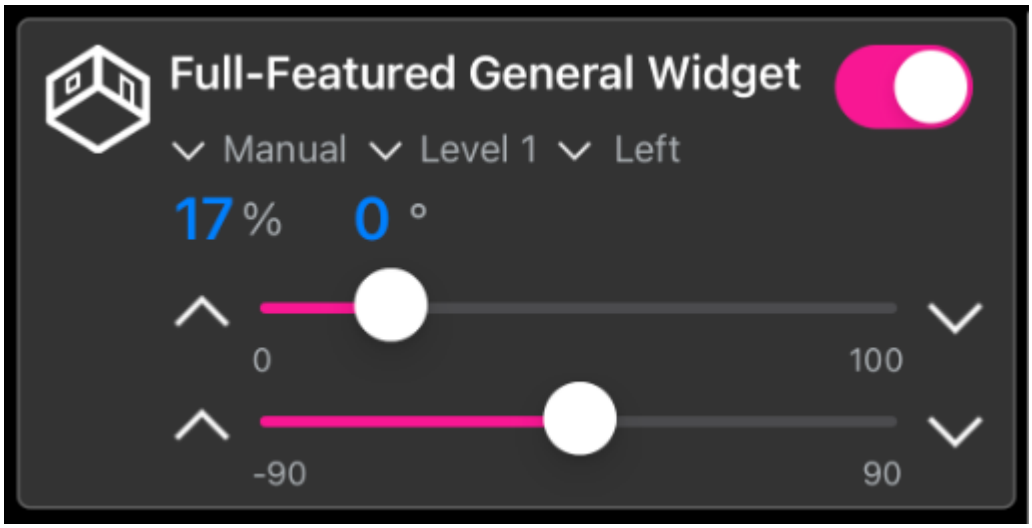
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: TimeSwitch.
iot.TimeSwitchStartTimeVisible	BOOL	Determines whether the start time is displayed (TRUE) or not (FALSE).
iot.TimeSwitchEndTimeVisible	BOOL	Determines whether the end time is displayed (TRUE) or not (FALSE).
iot.TimeSwitchStartDateVisible	BOOL	Determines whether the start date is displayed (TRUE) or not (FALSE).
iot.TimeSwitchEndDateVisible	BOOL	Determines whether the end date is displayed (TRUE) or not (FALSE).
iot.TimeSwitchDaysVisible	BOOL	Determines whether the weekdays are displayed (TRUE) or not (FALSE).
iot.TimeSwitchDateYearlyVisible	BOOL	Determines whether the attribute for the annual configuration is displayed (TRUE) or not (FALSE).
iot.TimeSwitchModeVisible	BOOL	Determines whether the mode is displayed (TRUE) or not (FALSE).
iot.TimeSwitchModeChangeable	BOOL	Determines whether the mode is adjustable (TRUE) or not (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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bOn	BOOL	Switches the timer on (TRUE) or off (FALSE).	Toggle switch top right.
tStartTime	TIME_OF_DAY	Start time of the timer.	Time on the left side of the arrow.
tEndTime	TIME_OF_DAY	End time of the timer.	Time on the right side of the arrow.
dStartDate	DATE	Start date of the timer.	Date on the left side of the arrow.
dEndDate	DATE	End date of the timer.	Date on the right side of the arrow.
bYearly	BOOL	Yearly.	Yearly (Depending on the language of the operating system).
bMonday	BOOL	Monday.	M.
bTuesday	BOOL	Tuesday.	D.
bWednesday	BOOL	Wednesday.	M.
bThursday	BOOL	Thursday.	D.
bFriday	BOOL	Friday.	F.
bSaturday	BOOL	Saturday.	S.
bSunday	BOOL	Sunday.	S.
sMode	STRING	Mode of the timer.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.8 Generic widget

The described widget is suitable for displaying a custom widget in the app, which is a flexible alternative in addition to the specific widgets. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

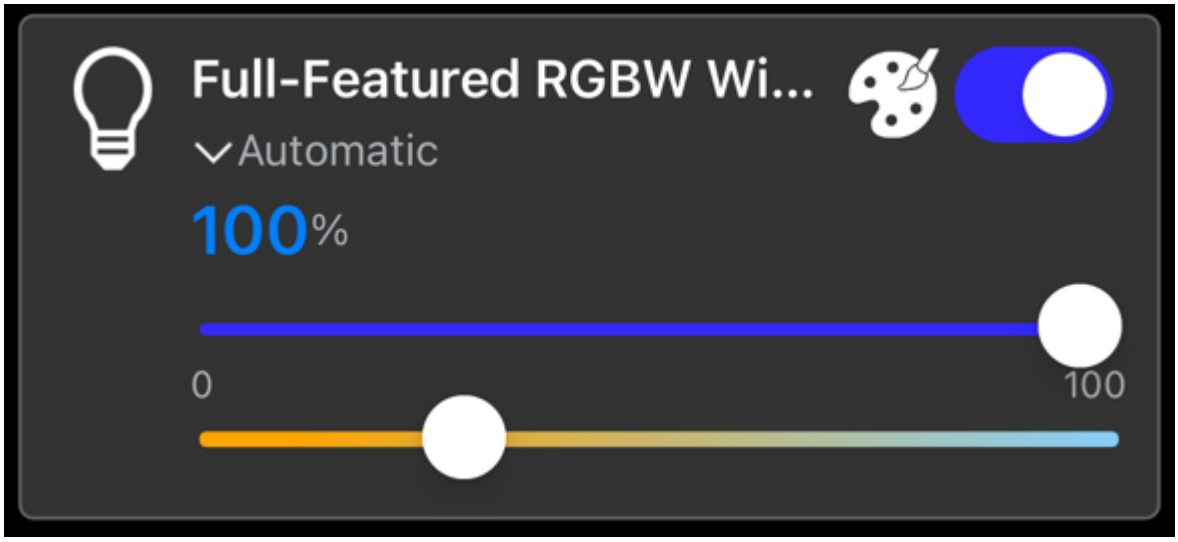
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: General.
iot.GeneralWidgetIcon	STRING	Determines which icon is used for the widget. The list of available icons can be found at List of available icons [▶ 80].
iot.GeneralWidgetColor	STRING	Determines with which color sliders and buttons are displayed. The list of available colors can be found at List of available colors [▶ 81]. The color must either be specified in one of the strings given in the appendix or as a hex code with "#" in front of the value.
iot.GeneralValue1SwitchVisible	BOOL	Determines whether the button is displayed (TRUE) or not (FALSE).
iot.GeneralValue2Visible	BOOL	Determines whether the left of the two numerical values is visible (TRUE) or not (FALSE).
iot.GeneralValue2SliderVisible	BOOL	Determines whether the upper of the two sliders is visible (TRUE) or not (FALSE).
iot.GeneralValue2SliderValuesVisible	BOOL	Determines whether the borders of the upper slider are visible (TRUE) or not (FALSE).
iot.GeneralValue2SliderButtonsVisible	BOOL	Determines whether the buttons of the upper slider are visible (TRUE) or not (FALSE).
iot.GeneralValue2SliderButtonsInverted	BOOL	This setting can be used to reverse the orientation of the top slider buttons.
iot.GeneralValue3Visible	BOOL	Determines whether the right of the two numerical values is visible (TRUE) or not (FALSE).
iot.GeneralValue3SliderVisible	BOOL	Determines whether the lower of the two sliders is visible (TRUE) or not (FALSE).
iot.GeneralValue3SliderValuesVisible	BOOL	Determines whether the borders of the lower slider are visible (TRUE) or not (FALSE).
iot.GeneralValue3SliderButtonsVisible	BOOL	Determines whether the buttons of the lower slider are visible (TRUE) or not (FALSE).
iot.GeneralValue3SliderButtonsInverted	BOOL	This setting can be used to reverse the orientation of the buttons of the lower slider.
iot.GeneralMode1Visible	BOOL	Determines whether the first mode is displayed (TRUE) or not (FALSE).
iot.GeneralMode1Changeable	BOOL	Determines whether the first mode is adjustable (TRUE) or not (FALSE).
iot.GeneralMode2Visible	BOOL	Determines whether the second mode is displayed (TRUE) or not (FALSE).
iot.GeneralMode2Changeable	BOOL	Determines whether the second mode is adjustable (TRUE) or not (FALSE).
iot.GeneralMode3Visible	BOOL	Determines whether the third mode is displayed (TRUE) or not (FALSE).
iot.GeneralMode3Changeable	BOOL	Determines whether the third mode is adjustable (TRUE) or not (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
```

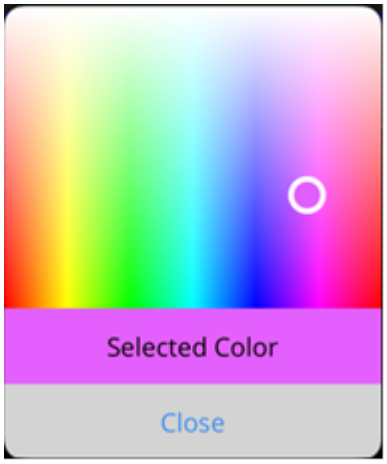
Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bValue1	BOOL	Turns the widget on (TRUE) or off (FALSE).	Toggle switch top right.
iot.Unit	STRING	Unit of the first value.	Unit after the first numerical value.
iot.MinValue	INT	Lower limit of the first value.	On the left side under the top slider.
iot.MaxValue	INT	Upper limit of the first value.	On the right side under the top slider.
nValue2	INT	First value.	Display in the left numerical value and additionally in the filling of the upper slider.
nValue2Request	INT	Request value for the first value.	The value to which the upper slider is moved.
nValue2Up	BOOL	One of the buttons for the top slider.	On the left side of the upper slider.
nValue2Down	BOOL	One of the buttons for the top slider.	On the right side of the top slider.
iot.Unit	STRING	Unit of the second value.	Unit after the second numerical value.
iot.MinValue	INT	Lower limit of the second value.	On the left side under the bottom slider.
iot.MaxValue	INT	Upper limit of the second value.	On the right side under the bottom slider.
nValue3	INT	Second value.	Display in the right numerical value and additionally in the filling of the lower slider.
nValue3Request	INT	Request value for the second value.	The value to which the lower slider is moved.
nValue3Up	BOOL	One of the buttons for the bottom slider.	On the left side of the bottom slider.
nValue3Down	BOOL	One of the buttons for the bottom slider.	On the right side of the bottom slider.
sMode1	STRING	First mode.	The currently displayed first mode.
aModes1	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user for the first mode.	By pressing on the current mode, the adjustable modes can be displayed.
sMode2	STRING	Second mode.	The currently displayed second mode.
aModes2	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user for the second mode.	By pressing on the current mode, the adjustable modes can be displayed.
sMode3	STRING	Third mode.	The currently displayed third mode.
aModes3	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user for the third mode.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.9 RGBW lighting

The described widget is suitable for operating RGBW lighting from the app. The various configuration options are described below. In the figure all available features of the widget are active.



The following illustration shows the color palette that becomes visible by clicking on the color palette icon. If the same value is selected for 100 ms when moving the selection point, it is sent by the app to the PLC.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

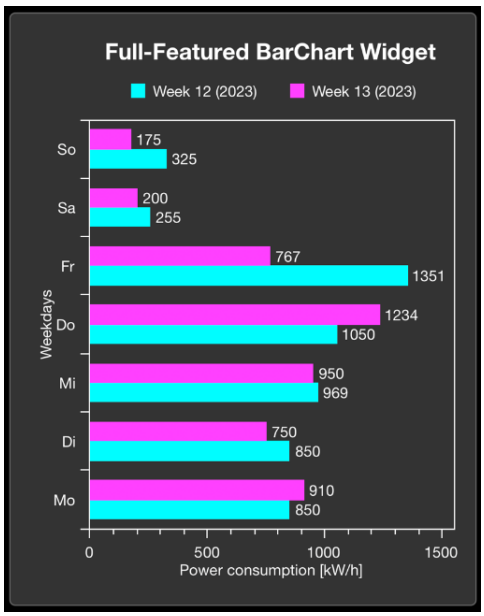
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: RGBW.
iot.LightValueVisible	BOOL	Determines whether the dimming value is displayed (TRUE) or not (FALSE).
iot.LightSliderVisible	BOOL	Determines whether the upper of the two sliders (for the dimming value of the light) is visible (TRUE) or not (FALSE).
ioT.LightColorPaletteVisible	BOOL	Determines whether the color palette is displayed (TRUE) or not (FALSE).
ioT.LightColorTemperatureSliderVisible	BOOL	Determines whether the lower of the two sliders (for the color temperature) is visible (TRUE) or not (FALSE).
iot.LightModeVisible	BOOL	Specifies whether the mode is displayed (TRUE) or not (FALSE).
iot.LightModeChangeable	BOOL	Specifies whether the mode is adjustable (TRUE) or not (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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Specifies the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bLight	BOOL	Switches the lighting on (TRUE) or off (FALSE).	Toggle switch top right.
iot.Unit	STRING	Unit of the dimming value.	Unit after the numerical value.
iot.MinValue	INT	Lower limit of the dimming value.	On the left side under the top slider.
iot.MaxValue	INT	Upper limit of the dimming value.	On the right side under the top slider.
nLight	INT	Dimming value of the lighting.	Numerical value and additional image in the filling of the upper slider.
nHueValue	INT	The Hue color value in the value range from 0 (red) to 360 (red again).	The number of degrees of the circle in the color palette.
nSaturation	INT	Saturation of the color value in the value range from 0 (grey) to 100 (selected color).	At the top of the color palette, the value is 0 and at the bottom of the color palette, the value is 100.
iot.MinValue	INT	Lower limit of the color temperature value.	No explicit display, defines the value range of the lower slider.
iot.MaxValue	INT	Upper limit of the color temperature value.	No explicit display, defines the value range of the lower slider.
nColorTemperature	INT	Value of the color temperature.	Display in the lower slider.
sMode	STRING	Mode of lighting.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.3.10 Bar chart

The widget described is suitable for displaying a horizontal bar chart in the app. This bar chart can display both individual bars and two bars to be compared. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{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;
```

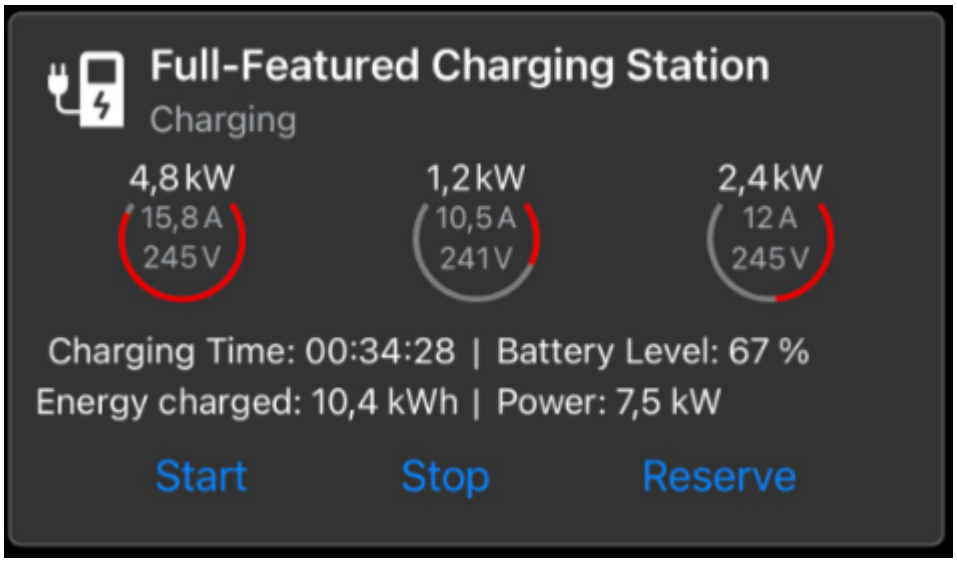
Attribute	Data type	Description
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: BarChart.
ioT.ChartXAxisLabel	STRING	The display name of the X-axis.
ioT.ChartYAxisLabel	STRING	The display name of the Y-axis.
iot.Unit	STRING	Unit displayed in square brackets after the display name of the X-axis.
ioT.ChartLegendVisible	BOOL	Determines whether the legend is displayed (TRUE) or not (FALSE).
ioT.ChartValuesVisible	BOOL	Determines whether the values of the bars are displayed in the diagram (TRUE) or not (FALSE).
iot.MinValue	INT	Lower limit of the value range of the diagram.
iot.MaxValue	INT	Upper limit of the value range of the diagram.
ioT.ChartBarColor1	STRING	Fixes the color of the upper bars. The list of available colors can be found at List of available colors [▶ 81]. The color must either be specified in one of the strings given in the appendix or as a hex code with "#" in front of the value.
ioT.ChartBarColor2	STRING	Fixes the color of the lower bars. The list of available colors can be found at List of available colors [▶ 81]. The color must either be specified in one of the strings given in the appendix or as a hex code with "#" in front of the value.

```
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
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
aDataSeries	ARRAY [0..n] OF INT	The first data series displayed in the bar chart. The array must be the same size as the ComparismDataSeries and the DataSeriesIdentifier.	Length of the upper beams.
aComparismDataSeries	ARRAY [0..n] OF INT	The second data series displayed in the bar chart. The array must be the same size as the DataSeries and the DataSeriesIdentifier. When sending an empty array, a bar chart with only one bar is displayed.	Length of the lower beams.
aDataSeriesIdentifier	ARRAY [0..n] OF STRING	Names of the values on the Y-axis. This array must be the same size as the DataSeries and the ComparismDataSeries.	Labels of the values on the Y-axis.
aLegendLabels	ARRAY [0..0] OF STRING ARRAY [0..1] OF STRING	Legend of the bar chart. The size of the array is determined by the number of bars displayed.	Between the title of the widget and the diagram.

5.3.11 Charging station

The widget described is suitable for displaying and operating a charging station in the app. The various configuration options are described below. In the figure all available features of the widget are active.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'ChargingStation'}
{attribute 'iot.ChargingStationReserveVisible' := 'true'}
```

```
{attribute 'iot.ChargingStationPhase2Visible' := 'true'}
{attribute 'iot.ChargingStationPhase3Visible' := 'true'}
stChargingStationWidgetSample : ST_ChargingStationWidgetSample;
```

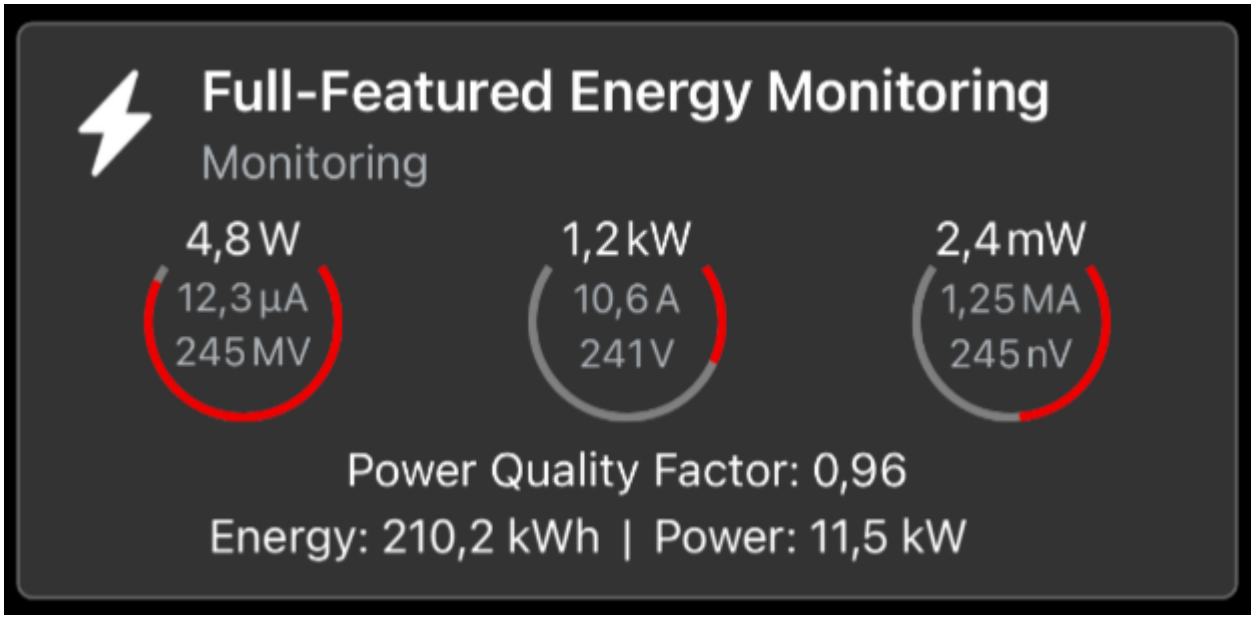
Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or additionally also write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: ChargingStation.
ioT.ChargingStationReserveVisible	BOOL	Specifies whether the reserve button is displayed (TRUE) or not (FALSE).
iot.ChargingStationPhase2Visible	BOOL	Specifies whether the values of the second phase are displayed (TRUE) or not (FALSE).
iot.ChargingStationPhase3Visible	BOOL	Specifies whether the values of the third phase are displayed (TRUE) or not (FALSE).

```
TYPE ST_ChargingStationWidgetSample :
STRUCT
    sDisplayName               : STRING;
    bStartCharging             : BOOL;
    bStopCharging              : BOOL;
    bReserveCharging           : BOOL;
    sStatus                    : STRING;
    nBatteryLevel              : UINT;
    nCurrentPower              : LREAL;
    aThreePhaseMaxPower        : ARRAY[0..2] OF LREAL;
    aThreePhaseCurrentPower    : ARRAY[0..2] OF LREAL;
    aThreePhaseAmperage        : ARRAY[0..2] OF LREAL;
    aThreePhaseVoltage         : ARRAY[0..2] OF LREAL;
    nChargingTime              : UDINT;
    nChargingEnergy            : LREAL;
END_STRUCT
END_TYPE
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bStartCharging	BOOL	Intended to trigger the start of a charging process.	The left button.
bStopCharging	BOOL	Intended to trigger the stop of a charging process.	The center button.
bReserveCharging	BOOL	Intended to trigger the reservation of a charging station.	The right button.
sStatus	STRING	Intended to indicate the status of charging.	Text under the display text of the widget.
nBatteryLevel	UINT	The charge status of the vehicle, if known.	Marked as "Battery Level" in this figure.
nCurrentPower	LREAL	The current power at which the vehicle is charging in kW.	Marked as "Power" in this figure.
aThreePhaseMaxPower	ARRAY [0..2] OF LREAL	The three maximum power values of the three phases in kW.	Scaling of the three red-filled circles.
aThreePhaseCurrentPower	ARRAY [0..2] OF LREAL	The current power of the three phases in kW.	Fill in the red circles and value above the circles.
aThreePhaseAmperage	ARRAY [0..2] OF LREAL	The current amperage of the three phases in A.	Upper value within the three circles.
aThreePhaseVoltage	ARRAY [0..2] OF LREAL	The current voltage of the three phases in V.	Lower value within the three circles.
nChargingTime	UDINT	The elapsed time of the charging process in seconds.	Marked as "Charging Time" in this figure.
nChargingEnergy	LREAL	The energy charged so far during the current charging process in kWh.	Marked as "Energy charged" in this figure.

5.3.12 Energy monitoring

The described widget is suitable for displaying the energy data in the app. The different configuration options are described below. In the figure all available features of the widget are active. The values shown in the widget do not claim to be realistic. They are merely intended to show different possibilities of presentation.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'EnergyMonitoring'}
{attribute 'iot.EnergyMonitoringPhase2Visible' := 'true'}
{attribute 'iot.EnergyMonitoringPhase3Visible' := 'true'}
stEnergyMonitoringWidgetSample : ST_EnergyMonitoringWidgetSample;
```

Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or additionally also write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: EnergyMonitoring.
ioT.EnergyMonitoringPhase2Visible	BOOL	Specifies whether the values of the second phase are displayed (TRUE) or not (FALSE).
ioT.EnergyMonitoringPhase3Visible	BOOL	Specifies whether the values of the third phase are displayed (TRUE) or not (FALSE).

```
TYPE ST_EnergyMonitoringWidgetSample :
STRUCT
    sDisplayName                : STRING;
    sStatus                     : STRING;
    aThreePhaseMaxPower         : ARRAY[0..2] OF LREAL;
    aThreePhaseCurrentPower     : ARRAY[0..2] OF LREAL;
    aThreePhasePowerUnits       : ARRAY[0..2] OF STRING;
    aThreePhaseAmperage         : ARRAY[0..2] OF LREAL;
    aThreePhaseAmperageUnits    : ARRAY[0..2] OF STRING;
    aThreePhaseVoltage          : ARRAY[0..2] OF LREAL;
    aThreePhaseVoltageUnits     : ARRAY[0..2] OF STRING;
    nPowerQualityFactor         : LREAL;
    nCurrentPower               : LREAL;
    sPowerUnit                  : STRING;
    nEnergy                     : LREAL;
    sEnergyUnit                 : STRING;
END_STRUCT
END_TYPE
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Determines the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
sStatus	STRING	Intended to indicate the status of energy monitoring.	Text under the display text of the widget.
aThreePhaseMaxPower	ARRAY [0..2] OF LREAL	The three maximum power values of the three phases.	Scaling of the three red-filled circles.
aThreePhaseCurrentPower	ARRAY [0..2] OF LREAL	The current power of the three phases.	Fill in the red circles and value above the circles.
aThreePhasePowerUnits	ARRAY [0..2] OF STRING	Intended to define the units of the three powers. You can find out more about processing the units later in this chapter.	Units behind the value above the circles.
aThreePhaseAmperage	ARRAY [0..2] OF LREAL	The three values of the amperages of the three phases.	Value of the upper number in the circles.
aThreePhaseAmperageUnits	ARRAY [0..2] OF STRING	Intended to define the units of the three amperages. You can find out more about processing the units later in this chapter.	Unit behind the upper value in the circles.
aThreePhaseVoltage	ARRAY [0..2] OF LREAL	The three values of the voltages of the three phases.	Value of the lower number in the circles.
aThreePhaseVoltageUnits	ARRAY [0..2] OF STRING	Intended to define the units of the three voltages. You can find out more about processing the units later in this chapter.	Unit behind the lower value in the circles.
nPowerQualityFactor	LREAL	The power quality factor.	Marked as "Power Quality Factor" in this figure.
nCurrentPower	LREAL	The current total power.	Marked as "Power" in this figure.
sPowerUnit	STRING	Unit of the current total power.	Behind the value for "Power".
nEnergy	LREAL	The energy charged so far during the current charging process.	Marked as “Energy” in this figure.
sEnergyUnit	STRING	Unit of the energy charged so far.	Behind the value for "Energy".

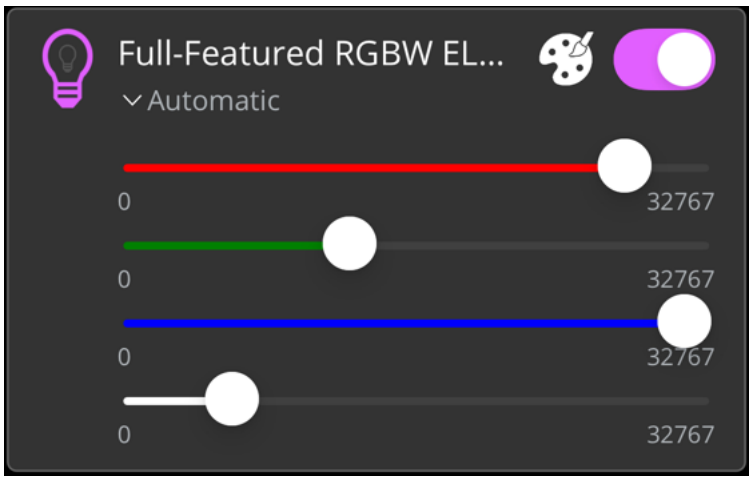
Automatic conversion of units

Starting from the base unit specified in the respective arrays, the current value is automatically converted in the range between nano (10^-9) and exa (10^18) to enable a meaningful display in the widget.

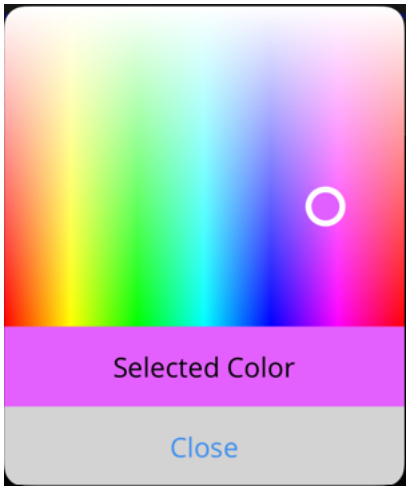
For example, if A is specified as the base unit, a value of 0.0001 A would be displayed as 100 µA. If the value is 10000 A, 10 kA would be displayed.

5.3.13 4-channel LED

The widget described is suitable for operating RGBW lighting from the app. Compared to the widget RGBW lighting [▶ 34], the focus here is specifically on the 4-channel EL2564. Separate values can be assigned to each individual channel, and a common color value can be set using the color palette. The various configuration options are described below. In the figure all available features of the widget are active.



The following illustration shows the color palette that becomes visible by clicking on the color palette icon.



The widget is transferred as a substructure in the overall structure of the SendData [▶ 55]() method. To build the widget, various PLC attributes are used when declaring the structure.

```
{attribute 'iot.ReadOnly' := 'false'}
{attribute 'iot.DisplayName' := 'Name for Widget'}
{attribute 'iot.WidgetType' := 'RGBWEL2564'}
{attribute 'iot.LedRedSliderVisible' := 'true'}
{attribute 'iot.LedGreenSliderVisible' := 'true'}
{attribute 'iot.LedBlueSliderVisible' := 'true'}
{attribute 'iot.LedWhiteSliderVisible' := 'true'}
{attribute 'iot.LedModeVisible' := 'true'}
{attribute 'iot.LedModeChangeable' := 'true'}
stGeneralWidgetSample : ST_RGBWEL2564WidgetSample;
```

Attribute	Data type	Description
iot.ReadOnly	BOOL	Determines whether the widget on the app side gets only read access (TRUE) or also additional write access to the PLC (FALSE).
iot.DisplayName	STRING	The display name of the widget in the app. This will be overwritten by sDisplayName as soon as sDisplayName is not an empty string.
iot.WidgetType	STRING	Type specification for the widget, in this case: RGBWEL2564.
iot.LedRedSliderVisible	BOOL	Specifies whether the slider is displayed for the red channel (TRUE) or not (FALSE).
iot.LedGreenSliderVisible	BOOL	Specifies whether the slider is displayed for the green channel (TRUE) or not (FALSE).
ioT.LedBlueSliderVisible	BOOL	Specifies whether the slider is displayed for the blue channel (TRUE) or not (FALSE).
ioT.LedWhiteSliderVisible	BOOL	Specifies whether the slider is displayed for the white channel (TRUE) or not (FALSE).
ioT.LedModeVisible	BOOL	Specifies whether the mode is displayed (TRUE) or not (FALSE).
ioT.LedModeChangeable	BOOL	Specifies whether the mode is adjustable (TRUE) or not (FALSE).

```
TYPE ST_RGBWEL2564WidgetSample :
STRUCT
    sDisplayName      : STRING := '';
    bOn               : BOOL := FALSE;
    nRed              : INT;
    nGreen            : INT;
    nBlue             : INT;
    nWhite            : INT;
    sMode             : STRING := 'Automatic';
    aModes            : ARRAY[0..1] OF STRING := ['Manual', 'Automatic'];
END_STRUCT
END_TYPE
```

Attribute	Data type	Description	Display in widget
sDisplayName	STRING	Specifies the display name of the widget and overwrites the PLC attribute 'iot.DisplayName'.	Display text of the widget.
bOn	BOOL	Switches the lighting on (TRUE) or off (FALSE).	Toggle switch top right.
nRed	INT	The value for the red channel in the value range from 0 to 32767.	Red colored slider.
nGreen	INT	The value for the green channel in the value range from 0 to 32767.	Green colored slider.
nBlue	INT	The value for the blue channel in the value range from 0 to 32767.	Blue colored slider.
nWhite	INT	The value for the white channel in the value range from 0 to 32767.	White colored slider.
sMode	STRING	Mode of lighting.	The currently displayed mode.
aModes	ARRAY [0..n] OF STRING	Array of the different modes that can be set by the user.	By pressing on the current mode, the adjustable modes can be displayed.

5.4 Nested structures

It is possible to communicate with the Communicator app from the PLC via several levels of nested structures. These nested structures can be directly displayed on the app side and can be expanded down to the last level.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17 or higher	App version 1.2.2 or higher	TC3_IotCommunicator

5.5 Limitation of decimal places

In many use cases it is sufficient in the app not to display all decimal places of floating-point numbers (PLC: REAL and LREAL). As an example, a temperature value is mentioned at this point, where a human being can still do something with a maximum of two digits after the decimal point.

At this point there are two possibilities to influence the number of decimal places displayed. In the first option, a setting is used for the entire app, and each variable is limited to a number of decimal places specified in the app settings (see App settings [▶ 69]). The second possibility is to set a certain number of decimal places for a single variable via the PLC attributes (cf. Limitation of decimal places [▶ 16]).

If different values for the number of decimal places are defined in the app settings and the setting on a single variable, the setting on the single variable is always taken into account first. It is therefore possible, for example, to define the value 2 for all floating-point numbers via the app settings and still deviate from this number for individual variables.

The restriction of decimal places means that the values are rounded. They are not cut off at all. The table below shows a simple example:

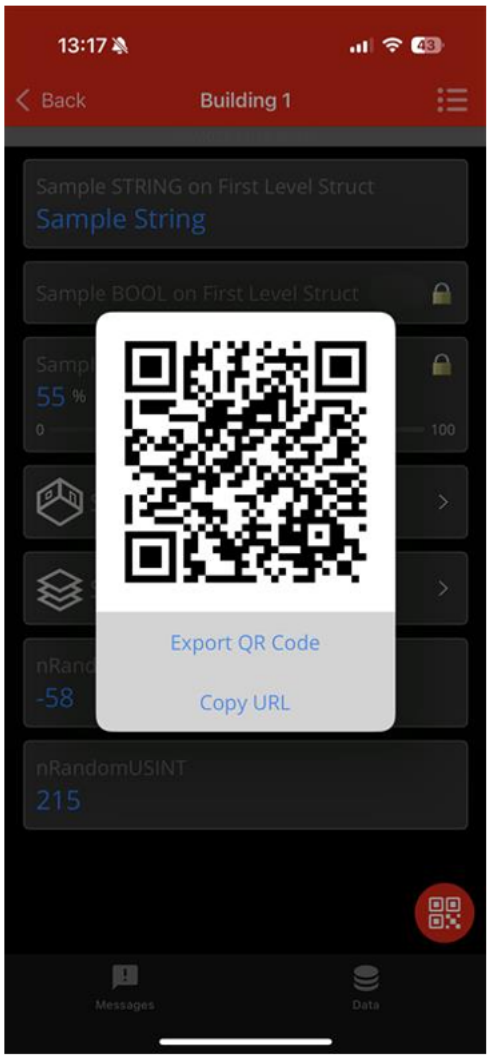
Value	Decimal Number Precision	Display in the app
1.68678	3	1.687
1.68678	1	1.7
1.68678	0	2

Requirements

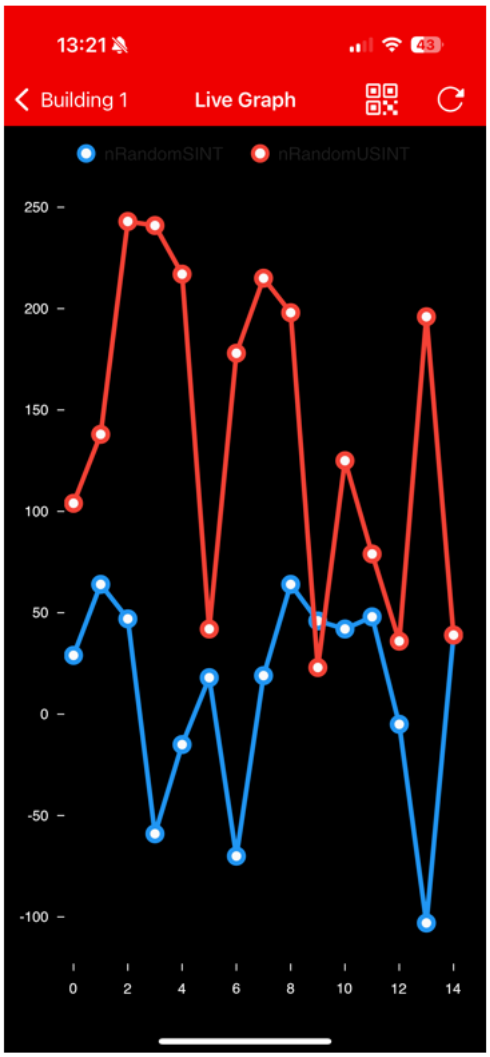
Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.23 or higher	App version 1.2.6 or higher	TC3_IotCommunicator

5.6 Navigation via QR code

A QR code can be generated for each page below a device. With the help of this QR code it is possible to open any page of the app directly. For this purpose, the QR code can be used as a graphic or the URL behind it can be used directly.



A QR code can also be generated for a graph. This QR code can be used to reopen this graph as well as a single page. For this purpose, a button for generating a QR code is displayed at the top of the navigation bar.



5.7 OnChange mechanisms

The OnChange mechanism is primarily intended for use with the SendDataAsString [▶ 56] mechanism. There are various references to the use of this mechanism.

Difference between the different methods

The SendData [▶ 55] method and the SendDataAsString [▶ 56] method are sent as retain messages. This has the effect that a newly connected app automatically has the most recently sent data available, even if no data is currently being sent.

If the respective OnChange methods are used, the messages are not sent as retain messages. With a new connection, the current status of the data should be sent as a retain message so that the newly connected app instance also has the current status of the data available. Further information on the OnChange methods can be found at SendData_OnChange [▶ 58] and SendDataAsString_OnChange [▶ 59].

Adding variables/widgets

If the SendDataAsString [▶ 56] mechanism is used, a certain structure of the JSON document must be taken into account. If changes (add/delete or adjust) are made to the structure between two calls, the following points must be observed:

- If a variable/widget is added and no change to the order is desired, the ForceUpdate parameter can remain FALSE.
- If a variable/widget is added and a change to the order is desired, the ForceUpdate parameter must be set to TRUE for a call. It should then be set to FALSE again.

- If a variable/widget is deleted, the page in the app must be closed and reopened once.
- If a variable/widget is changed in relation to the metadata (e.g. reconfiguration of a widget), ForceUpdate must be set to TRUE for a call, otherwise only the values of the widget are updated. It should then be set to FALSE again.

Additional Notes

- The bNewAppSubscribe parameter of the FB_IotCommunicator [▶ 52] function block goes TRUE for one cycle when a new app instance connects to the topic on the message broker. This parameter can be used to identify the right time to send a retain message.
- The nActiveAppInstances parameter of the FB_IotCommunicator [▶ 52] function block specifies the number of connected app instances. If no app is connected, no data is sent in order to save performance and data volume.

Structure TwinCAT JSON

The TwinCAT IoT Communicator product range uses a JSON format called TwinCAT JSON for communication: the structure of a TwinCAT JSON document is described below using the widgets Socket [▶ 22] and Ventilation [▶ 25] as an example.

```
{
    "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
}
```

Range	Description
Timestamp	Must contain a timestamp per message in the format: "YYYY-MM-DDThh:mm:ss.fff" e.g. "2022-08-04T07:15:06.176".
GroupName	Name of the entry node of the Communicator function block in the app.
Values	The values to be displayed, starting on the first page, with subsequent nesting.
MetaData	Everything that is implemented in PLC attributes (for example, the configuration of widgets).
ForceUpdate	Optional parameter. Is used in the OnChange mechanism to trigger an update after changes. More accurate information can be found at OnChange mechanisms [▶ 48].

5.8 Using UTF-8 characters

Variables

TwinCAT uses the ISO/IEC 8859-1 character set by default. The TwinCAT IoT Communicator app, on the other hand, uses UTF-8 for decoding STRINGs.

The first 128 characters are the same for UTF-8 and ISO/IEC 8859-1. For all characters not contained in the first 128 characters, the following method must be applied to the STRING variable:

```
sMyUTF8Text : STRING := wsLiteral_TO_UTF8( "äöüßéèêµ€° Αθηναι İstanbul Київ");
```

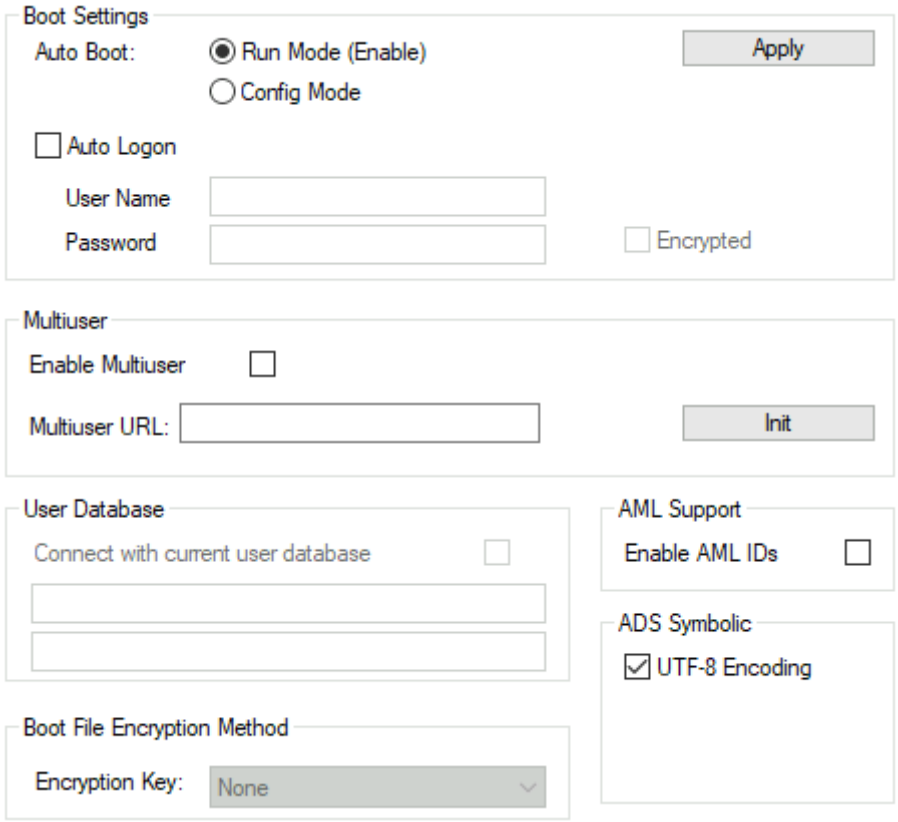
When used in arrays, the conversion must take place outside the array:

```
aData : ARRAY[0..2] OF STRING := ['Sample1', sMyUTF8Text, 'Sample2'];
```

It should be noted at this point that this conversion must also be taken into account in other program sequences.

Attributes

If special characters are to be used in the attributes, a check mark can be set for the UTF-8 encoding of the ADS symbols in TwinCAT Build 3.1.4024.



With TwinCAT 3.1.4026, this setting is set by default. Therefore, it no longer needs to be set manually by the user.

5.9 User authorizations

It is possible to assign authorizations for users. This feature can be used to display individual areas per user or to deny certain users access to areas of the app.

This feature is only intended for display within the app. If there is a direct connection to the message broker, all content can potentially be viewed.

Configuration at device level

It is possible to assign user authorization for an entire device (and thus an instance of the function block FB_IotCommunicator [▶ 52]). The variable *sPermittedUsers* is available in FB_IotCommunicator [▶ 52] for this purpose.

Configuration at variable level

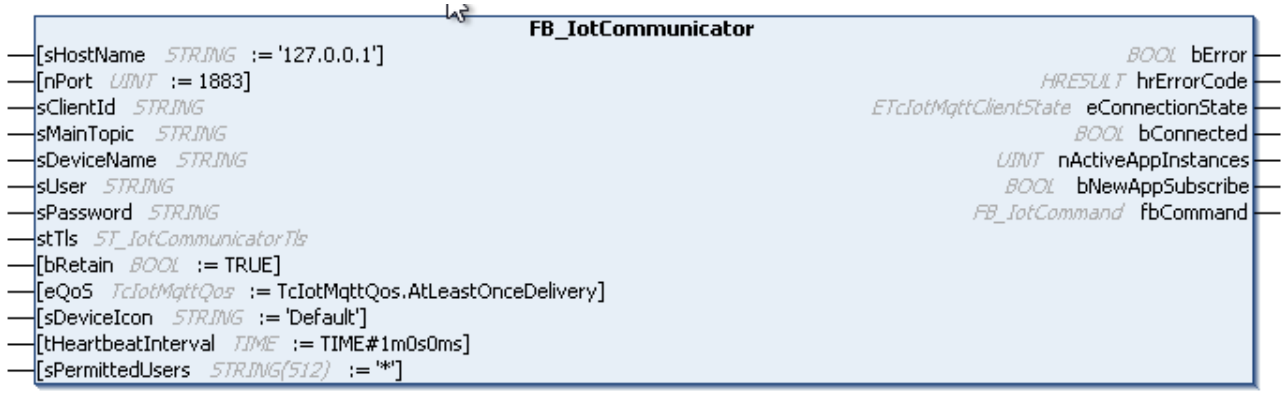
It is also possible to assign authorizations for each variable. The *iot.PermittedUsers* attribute can be used for this purpose. This applies to both structures and individual variables.

At both device and variable level, the permitted users are specified comma-separated by a string. Further information can be found at FB_IotCommunicator [▶ 52] and Attributes [▶ 15].

6 PLC API

6.1 Function blocks

6.1.1 FB_IotCommunicator



The function block enables communication with an MQTT broker.

An FB_IotCommunicator function block deals with the connection to precisely one broker and with sending and receiving of data for precisely one device. To ensure the background communication to this broker and thus enable sending and receiving of data and messages, the Execute method of the function block must be called cyclically.

All connection parameters exist as input parameters and are evaluated when a connection is established.

Syntax

Definition:

```
FUNCTION_BLOCK FB_IotCommunicator
VAR_INPUT
    sHostName           : STRING := '127.0.0.1';
    nPort               : UINT := 1883;
    sClientId           : STRING;
    sMainTopic          : STRING;
    sDeviceName         : STRING;
    sUser               : STRING;
    sPassword           : STRING;
    stTls               : ST_IotCommunicatorTls;
    bRetain             : BOOL := TRUE;
    eQoS                : TcIotMqttQos := TcIotMqttQos.AtLeastOnceDelivery;
    sDeviceIcon         : STRING;
    tHeartbeatInterval  : TIME := TIME#1m0s0ms;
    sPermittedUsers     : STRING(512) := '*';
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    hrErrorCode         : HRESULT;
    eConnectionState    : ETcIotMqttClientState;
    bConnected          : BOOL;
    nActiveAppInstances : UINT;
    bNewAppSubscribe    : BOOL;
    fbCommand           : FB_IoTCommand;
END_VAR
```

Inputs

Name	Type	Description
sHostName	STRING	sHostName can be specified as the host name or as the IP address. If no information is provided, the local host is used.
nPort	UINT	The host port is specified here. (Default: 1883)
sClientId	STRING	The client ID can be specified individually. If no ID is specified, it is generated.
sMainTopic	STRING	Here you specify the main topic in which the data and messages are sent.
sDeviceName	STRING	Here you can enter the name of the device to which the data and messages belong.
sUser	STRING	Optionally, a user name can be specified.
sPassword	STRING	A password for the user name can be entered here.
stTLS	ST_IotCommunicatorTls [▶ 61]	Parameter structure If the broker offers a TLS-secured connection, the required configuration can be implemented here.
bRetain	BOOL	By default, the broker stores the current data and the last 255 messages, together with the current device status. If this is not desirable, bRetain can be set to FALSE.
eQoS	TcIotMqttQos	The Quality of Service (QoS for short) can be set with this setting.
sDeviceIcon	STRING	This setting can be used to change the icon of the Communicator function block. If the setting is not set, the TwinCAT CD is used as icon by default. The list of available icons can be found at List of available icons [▶ 80].
tHeartbeatInterval	TIME	The timespan after which the online information of this function block is updated. The online information is visible at the top level in the app, where you can see the overview of the individual function blocks. The default value is 60 seconds.
sPermittedUsers	STRING(512)	This variable can be used to specify the users who are authorized to see a device. The users are separated by a comma. With the default value '*', all users are authorized to see the device.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error situation occurs.
hrErrorCode	HRESULT	Returns an error code if the bError output is set.
eConnectionState	ETcIotMqttClientState	Indicates the state of the connection between client and broker as enumeration ETcIotMqttClientState.
bConnected	BOOL	TRUE if there is a connection between the client and the broker.
nActiveAppInstances	UINT	Shows the number of currently connected app instances.
bNewAppSubscribe	BOOL	Goes TRUE for one cycle when a new app instance has connected. Can be used when using OnChange functionalities to publish the current status of the data again as a retain message during the connection.
fbCommand	FB_IotCommand [▶ 59]	Provides all the necessary functionality to evaluate received data ("Commands").

Methods

Name	Description
Execute [▶ 55]	Method for background communication with the TwinCAT driver. This method must be called cyclically.
SendData [▶ 55]	Method for sending data to the specified MQTT message broker as a retain message.
SendMessage [▶ 56]	Method for sending a (push) message to the specified MQTT message broker.
SendDataAsString [▶ 56]	Method for sending data to the specified MQTT message broker where the JSON document is transferred directly.
SendData_OnChange [▶ 58]	Method for sending OnChange data to the specified MQTT message broker.
SendDataAsString_OnChange [▶ 59]	Method for sending OnChange data to the specified MQTT message broker where the JSON document is transferred directly.

Strings in UTF-8 format

The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication.

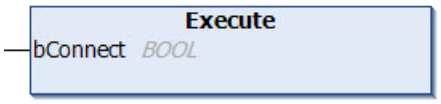
In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotCommunicator library is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.0	IPC or CX (x86, x64, Arm®)	Tc3_IotCommunicator

6.1.1.1 Execute



This method must be called cyclically in order to ensure the background communication with the MQTT broker.

Syntax

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

Inputs

Name	Type	Description
bConnect	BOOL	The connection to the broker is established when bConnect is set to TRUE. bConnect must remain set to maintain the connection. The connection to the broker is cut off by calling the Execute() method with FALSE as the input.

Any errors are reported at the outputs bError, hrErrorCode and eConnectionState of the function block instance.

6.1.1.2 SendData

This method is called once to send data to the broker.

Syntax

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

Return value

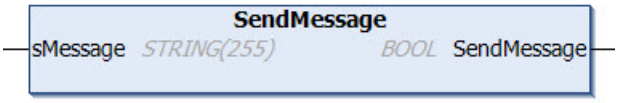
Name	Type	Description
SendData	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
pMachineStruct	PVOID	Address for the structure in which the device variables are declared.
nStructSize	UINT	Size of the structure specified in pMachineStruct.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

6.1.1.3 SendMessage



This method is called once to send a (push) message to the broker. This message is then displayed in the app. This is not a message that is visible as a push message on the mobile phone.

Syntax

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

Return value

Name	Type	Description
SendMessage	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
sMessage	STRING	Text of the (push) message to be sent to the broker.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

Strings in UTF-8 format

The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication.

In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotCommunicator library is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

6.1.1.4 SendDataAsString

This method is called once to send data to the broker. Unlike the classic SendData [▶ 55]() method, the JSON document is passed directly at this point. In this way, the user achieves greater flexibility, but in return must send a correctly constructed JSON document to the app.

The method is intended for users experienced in handling JSON documents. In case of incorrectly formatted JSON documents, the information cannot be displayed in the app.

Syntax

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

Return value

Name	Type	Description
SendDataAsString	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
sJsonString	POINTER TO STRING	Pointer to the JSON string to be sent.
nJsonLen	UINT	Length of the JSON string

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

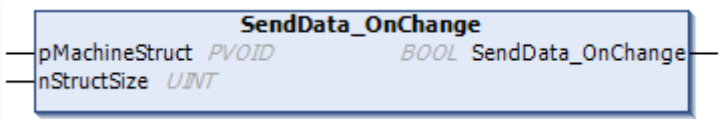
Structure TwinCAT JSON

The TwinCAT IoT Communicator product range uses a JSON format called TwinCAT JSON for communication: the structure of a TwinCAT JSON document is described below using the widgets Socket [▶ 22] and Ventilation [▶ 25] as an example.

```
{
    "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
}
```

Range	Description
Timestamp	Must contain a timestamp per message in the format: "YYYY-MM-DDThh:mm:ss.fff" e.g. "2022-08-04T07:15:06.176".
GroupName	Name of the entry node of the Communicator function block in the app.
Values	The values to be displayed, starting on the first page, with subsequent nesting.
MetaData	Everything that is implemented in PLC attributes (for example, the configuration of widgets).
ForceUpdate	Optional parameter. Is used in the OnChange mechanism to trigger an update after changes. More accurate information can be found at OnChange mechanisms [▶ 48].

6.1.1.5 SendData_OnChange



This method is called once to send data to the broker.

The SendData [▶ 55] method always transfers the entire data as a retain message. With the SendData_OnChange method, on the other hand, it is possible to transfer individual parts of the data in order to save data traffic. However, it should be noted that with the SendData_OnChange method the messages are not sent as retain messages and are therefore only known to currently connected app instances. The data must be kept ready in the TwinCAT project and sent as a complete data block and as a retain message when a new client is connected using the SendData [▶ 55] method.

It should be noted at this point that the OnChange mechanisms [▶ 48] are mainly intended for use with the SendDataAsString [▶ 56] methods. With the SendData methods, OnChange can only be used if a structure is used when sending the structure that has the same variable path up to the variables to be changed.

Syntax

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

Return value

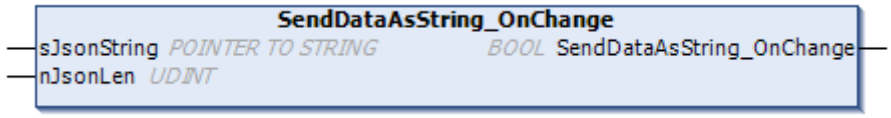
Name	Type	Description
SendData_OnChange	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
pMachineStruct	PVOID	Address for the structure in which the device variables are declared.
nStructSize	UINT	Size of the structure specified in pMachineStruct.

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

6.1.1.6 SendDataAsString_OnChange



This method is called once to send data to the broker. Unlike the classic SendData [▶ 55]() method, the JSON document is passed directly at this point. In this way, the user achieves greater flexibility, but in return must send a correctly constructed JSON document to the app.

The method is intended for users experienced in handling JSON documents. In case of incorrectly formatted JSON documents, the information cannot be displayed in the app.

The SendDataAsString [▶ 56] method transmits the data as a retain message. With the SendDataAsString_OnChange method, on the other hand, it is possible to transfer individual parts of the data in order to save data traffic. However, it should be noted that with the SendDataAsString_OnChange method the messages are not sent as retain messages and are therefore only known to currently connected app instances. The data must be kept ready in the TwinCAT project and sent as a complete data block when a new client is connected using the SendDataAsString [▶ 56] method.

Syntax

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

Return value

Name	Type	Description
SendDataAsString_OnChange	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
sJsonString	POINTER TO STRING	Pointer to the JSON string to be sent.
nJsonLen	UINT	Length of the JSON string

Possible errors are output at the outputs bError and hrErrorCode of the function block instance.

6.1.2 FB_IotCommand

The function block FB_IotCommand provides functions for evaluating received commands. It must not instantiated, since it is already declared at the output of the FB_IotCommunicator [▶ 52] instance and is used to access the outputs and received commands.

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
```

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error situation occurs.
hrErrorCode	HRESULT	Returns an error code if the bError output is set.
bAvailable	BOOL	TRUE, if a new command is available.
sVarName	STRING	If bAvailable is TRUE, sVarName contains the name of the variable that was received.

Methods

Name	Description
GetValue [▶ 60]	Method for accessing the value of the command, if bAvailable is TRUE
Remove [▶ 61]	Method for discarding the currently available command

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.0	IPC or CX (x86, x64, Arm®)	Tc3_IotCommunicator

6.1.2.1 GetValue

This method is called to access the value of the variable in the current command.

Syntax

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

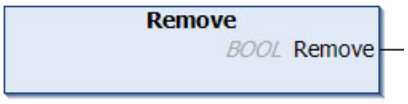
Return value

Name	Type	Description
GetValue	BOOL	The method returns the return value TRUE if the call was successful.

Inputs

Name	Type	Description
pValue	PVOID	Address of the variable to which the received value is to be written.
nSize	UDINT	Size of the variable specified in pValue
eDatatype	E_IotCommunicatorDatatype	Data type of the variable specified in pValue, based on enum E_IotCommunicatorDatatype

6.1.2.2 Remove



This method is called to remove the currently available command from the memory.

Return value

Name	Type	Description
Remove	BOOL	This value is set to TRUE if the method is called successfully.

6.2 Data types

6.2.1 ST_IotCommunicatorTls

TLS security settings for the 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	Type	Description
eVersion	E_IotCommunicatorTlsVersion	TLS version to be used, based on enum E_IotCommunicatorTlsVersion.
sCA	STRING(255)	Certificate of the certificate authority (CA)
sCert	STRING(255)	Client certificate that is used for authentication at the broker (optional)
sKeyFile	STRING(255)	Private key of the client
sKeyPwd	STRING(255)	Password of the private key, if applicable
bNoServerCertCheck	BOOL	Disables verification of the server certificate validity. If communication is to take place without TLS encryption (HTTP), this value must remain FALSE.

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	Type	Description
type_STRING	INT	Data type STRING.
type_BOOL	INT	Data type BOOL.
type_SINT	INT	Data type SINT.
type_INT	INT	Data type INT.
type_DINT	INT	Data type DINT.
type_LINT	INT	Data type LINT.
type_USINT_BYTE	INT	Data type USINT_BYTE.
type_UINT_WORD	INT	Data type UINT_WORD.
type_UDINT_DWORD	INT	Data type UDINT_DWORD.
type_ULINT	INT	Data type ULINT.
type_REAL	INT	Data type REAL.
type_LREAL	INT	Data type LREAL.

6.2.3 E_IotCommunicatorTlsVersion

Syntax

```
TYPE E_IotCommunicatorTlsVersion :
{
    tlsv1  :=0,
    tlsv1_1:=1,
```

```
    tlsv1_2:=2
) INT;
END_TYPE
```

Parameter

Name	Type	Description
tlsv1	INT	TLS version 1.
tlsv1_1	INT	TLS version 1.1.
tlsv1_2	INT	TLS version 1.2.

7 App

The TwinCAT IoT Communicator app can be downloaded free of charge from the Apple AppStore or Google PlayStore.

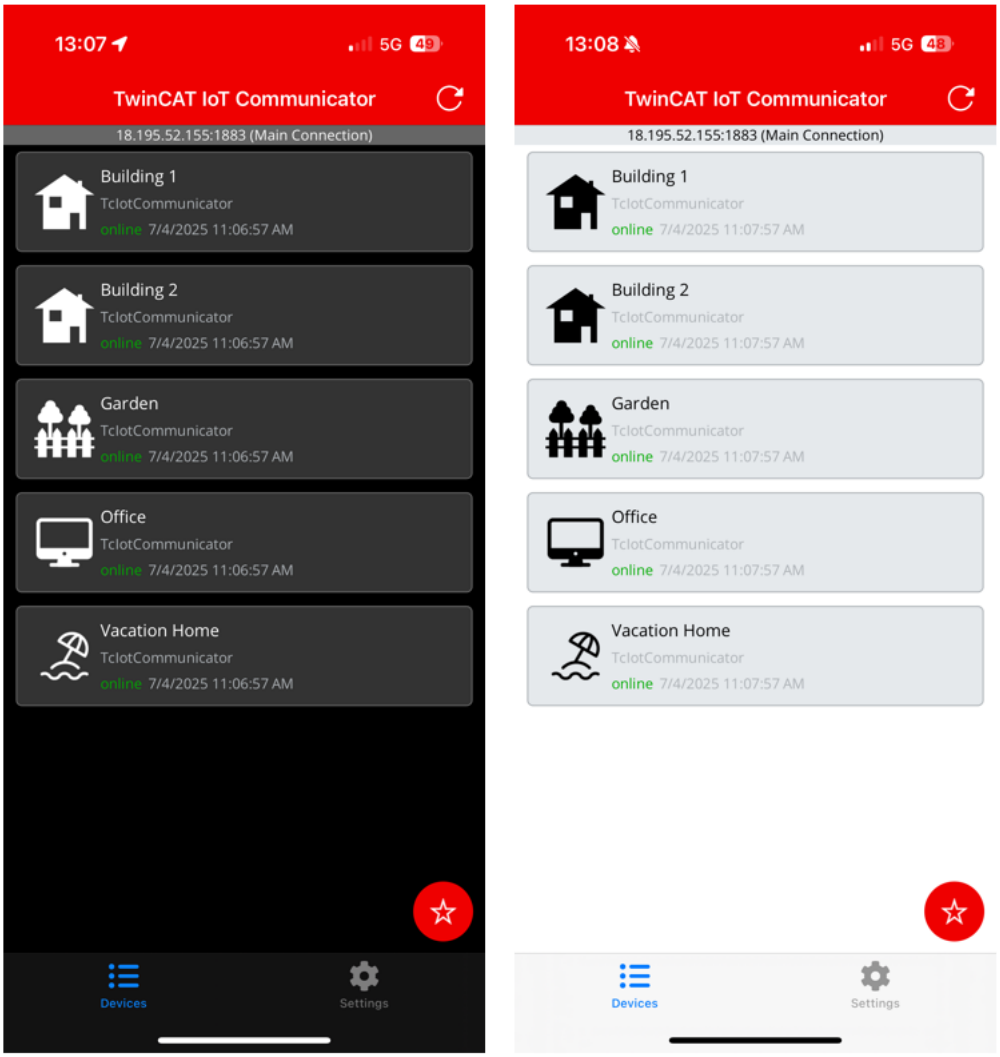
Download on the App Store

GET IT ON Google Play

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

At this point it is recommended in the case of new projects to work with the latest versions of both the app and TwinCAT as the engineering so that new features can also be used.

Since the app version 1.2.2, the "dark mode" has also been supported in addition to the "light mode". The mode in which the app is displayed depends on the respective operating system settings.

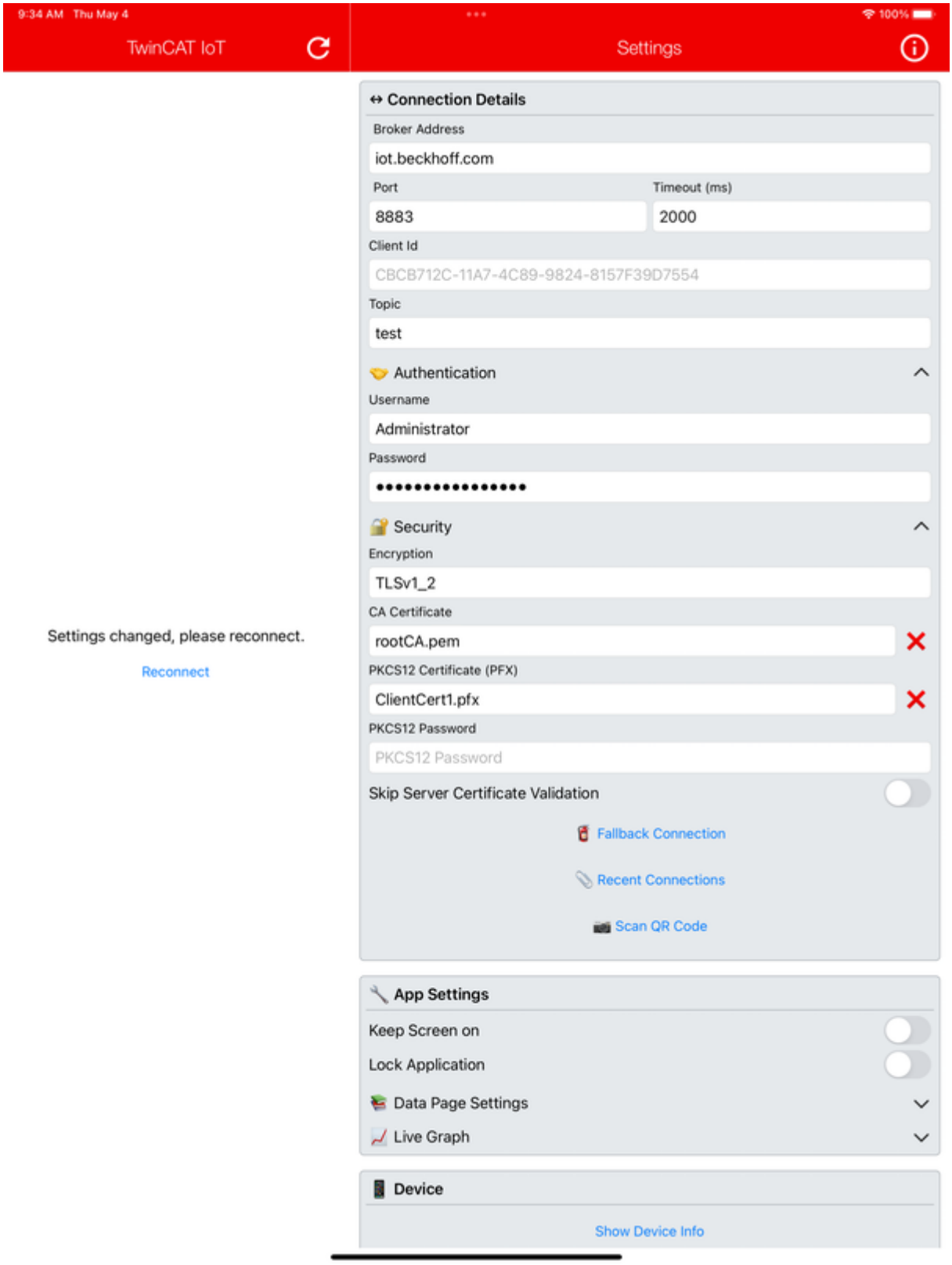


7.1 Settings

The settings within the app are divided into three different areas. The first area enables the configuration of the connection to the broker, general app settings can be made in the second area and device information can be displayed in the third and last area.

7.1.1 Connection settings

The app and the PLC must be connected to the same message broker in order to be able to receive data from the PLC. The different setting options for connecting to this message broker are described below.



Basic settings

Setting	Meaning
Broker Address	IP address or host name of the message broker.
Port	The port of the message broker. Usually 1883 (MQTT) or 8883 (MQTT TLS).
Timeout	This setting specifies the time after the connection to the message broker runs into a timeout. After this time, the connection to the second message broker is attempted if a fallback connection is active.
Client Id	The client ID of the app with which the connection to the message broker is established. If no user-defined value is entered, the unique device identifier of the mobile device is used.
Topic	Main topic via which the messages from the associated PLC program are communicated.

Authentication

Depending on the broker configuration, it may be necessary to enter a user name and password when establishing the connection. If a broker with the option of anonymous access is used, these boxes in the configuration are left empty.

Setting	Meaning
User name	User name for logging into the message broker
Password	Password associated with the user

Security

In addition to the authentication, the encryption of messages plays an important role.

Setting	Meaning
Encryption	Selection of the encryption protocol.
CA certificate	Referencing the CA certificate as a file. Free file access with Android, only in the “TwinCAT IoT” area under "On my iPhone" with iOS. Further information under Installation of CA certificates [▶ 69].
PKCS12 Certificate (PFX)	Referencing the client certificate as a file. Free file access with Android, only in the “TwinCAT IoT” area under "On my iPhone" with iOS. The certificate must be available as a PFX file. Information on the conversion can be found in popular technical literature.
PKCS12 Password	Password for the PFX file.
Skip Server Certificate Validation	This setting disables the validation of the server certificate.

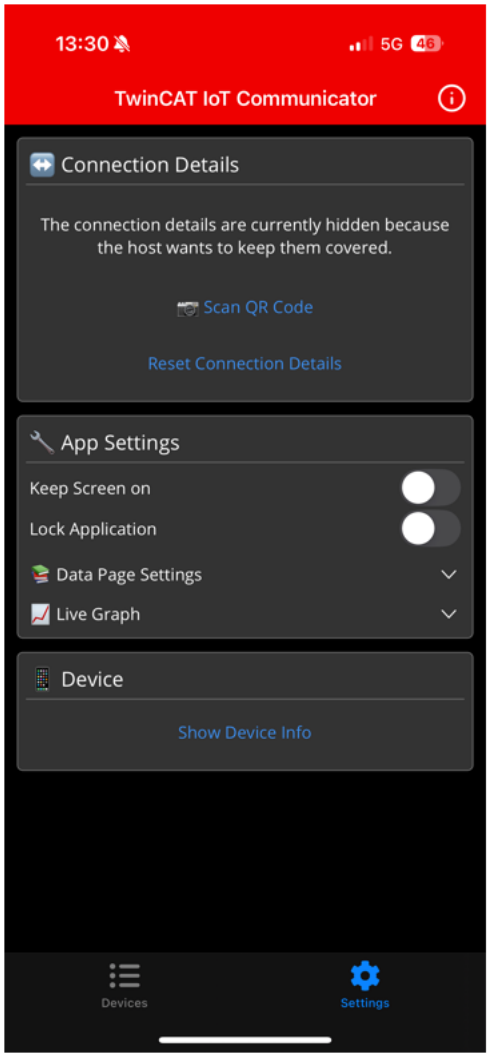
Advanced Settings

Setting	Meaning
Fallback Connection	An alternative connection to another message broker can be specified here if the primary connection cannot be reached. After the timeout defined above in the settings, the connection to the fallback connection is attempted.
Recent Connections	The most recent configured connections are displayed here. The connection parameters are inserted automatically by clicking the individual boxes. A new entry is added in the case of a new connection attempt. If a connection has already been established before with these parameters, the entry is placed at the top of the list.
Scan QR code	A QR code with the connection parameters can be scanned here. The formatting can be found below in a separate section.

Use of QR codes for establishing a connection

The settings page in the app offers an option to scan a QR code containing the connection parameters. The selection options with regard to security are less extensive in comparison with a manual connection setting. In addition, the user should consider that anyone can gain access via the QR code, depending on the location.

It is possible to equip a QR code with the so-called lock parameter. In this case, a connection can be established via the QR code, but no connection details are displayed to the app user. It should be noted at this point that a user can access the data by directly reading the QR code.



The content of an example QR code looks like this:

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

The connection parameters for the broker address and the broker port as well as for the topic "TOPICNAME" are entered here by scanning the QR code. The following list describes the possible parameters that can be mapped via the URL:

Parameter	Values
broker	IP address or host name of the broker.
port	Port of the message broker (normally 1883 or 8883).
clientid	Client ID of the app, if required.
topic	Topic to which the Communicator PLC library is published.
user	User name for logging into the message broker.
password	Password to log in to the message broker.
lock	The possible values are "true" or "1" if you want to hide the connection details.
tls	Possible values: "Default", "TLSv1_0", "TLSv1_1", "TLSv1_2". **No paths to certificates can be specified via the QR code. TLS can therefore only be used without a client certificate and by skipping the validation of the server certificate. As a result, encryption is achieved, but this does not guarantee any security in communication!**
skipCertCheck	The possible values are "true" or "false" to enable or disable the validation of the server certificate. Further description in the "tls" line.

7.1.1.1 Installation of CA certificates

To reference CA certificates under the two supported operating systems, Android and iOS, they must first be installed. You will find brief instructions for both below, for further information please refer to the documentation of the two manufacturers.

Installation of CA certificates Android

1. Select the path: Settings/Security/Encryption and credentials/Install a certificate/CA certificate.
2. Select a CA certificate.
3. Assign a name for the certificate (optional).

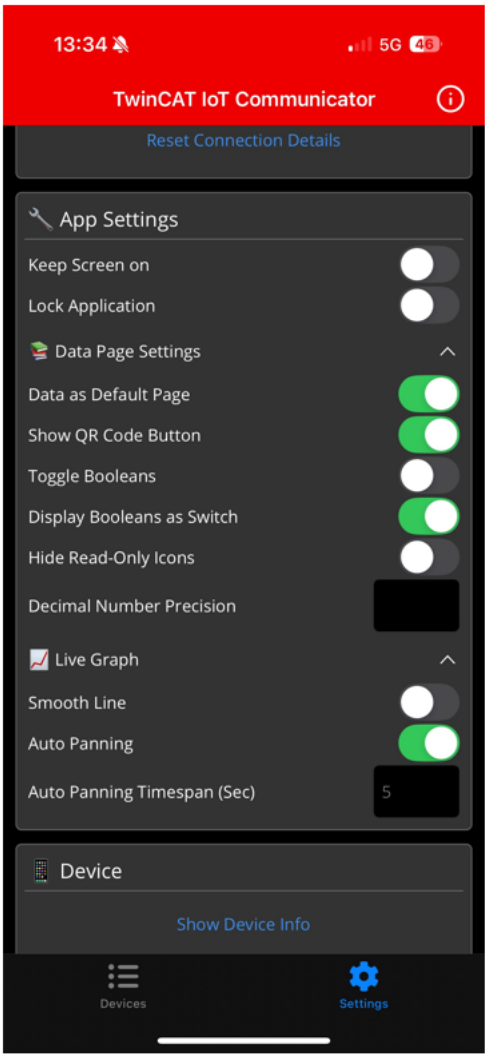
⇨ The system reports the successful installation of the CA certificate.

Installation of CA certificates iOS

1. Click on the CA certificate file (must be located outside the TwinCAT IoT folder). This loads the profile.
2. Select the path: Settings/General/Profile.
3. Choose the profile of the CA certificate.
4. Press **Install** in the top right-hand corner.
5. Confirm the installation by entering the code.
6. Read the warnings at the top right and then press **Install** to confirm.
7. Select the path: Settings/General/Info/Certificate trust settings.
8. Activate the installed certificate under **Activate full trust for root certificates**

7.1.2 App settings

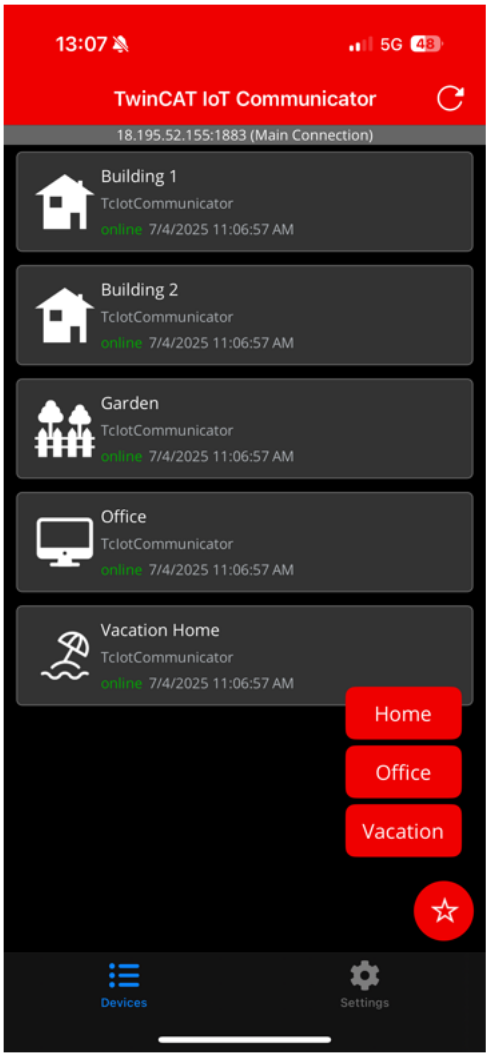
In addition to the connection settings, general settings can be made for the app. These are described below.


13:34
5G
TwinCAT IoT Communicator
Reset Connection Details
App Settings
Keep Screen on
Lock Application
Data Page Settings
Data as Default Page
Show QR Code Button
Toggle Booleans
Display Booleans as Switch
Hide Read-Only Icons
Decimal Number Precision
Live Graph
Smooth Line
Auto Panning
Auto Panning Timespan (Sec)
5
Device
Show Device Info
Devices
Settings

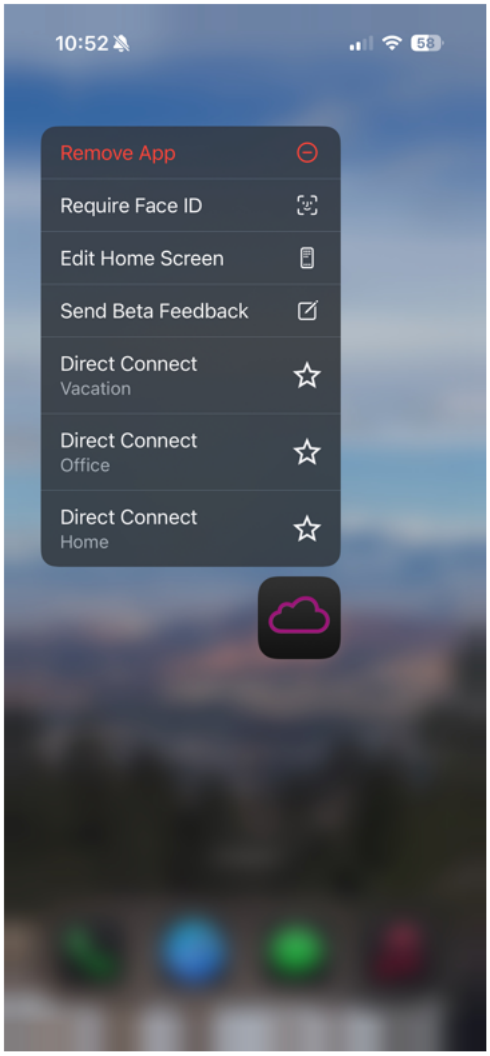
Setting	Meaning	Default
Keep Screen on	When activated, the mobile device screen will not turn off while the app is open.	FALSE
Lock Application	If activated, the set secure route of the operating system is used in order to protect the app against unauthorized access (Face-ID, Touch-ID, Code, etc.)	FALSE
Data as Default Page	If activated, the Data tab is opened when opening a device, otherwise the Messages tab is opened.	TRUE
Show QR Code Button	A button for generating a QR code is displayed on each page within a Device.	TRUE
Toggle Booleans	If this setting is activated, Boolean variables can be switched directly by pressing the display field. If inactive, a selection dialog opens on pressing the variable. This setting is only relevant if "Display Booleans as Switch" is FALSE.	FALSE
Display Booleans as Switch	Boolean variables are displayed as switches. If this setting is deactivated, these variables are displayed as a text field.	TRUE
Hide Read-Only Icons	When activated, the lock icon for displaying read-only elements is hidden. **There is then no way to recognize a read-only element.**	FALSE
Decimal Number Precision	Describes the number of decimal places to which REAL or LREAL values are rounded when displayed in the app. If a different value is defined on the PLC side for a single variable, the Decimal Number Precision for that variable is overwritten from the PLC.	This value is not set by default.
Smooth Line	When displaying a live graph, the curve of the graph is shown rounded off if this feature is activated.	FALSE
Auto Panning	If activated, the graph is trimmed to a certain timespan.	TRUE
Auto Panning Timespan	The timespan to which a graph is trimmed in the Live View.	5s

7.1.3 Favorites

The connection settings contain an overview of the most recent connections. Click on **Recent connections** to open them. In this overview, the user can assign names for a connection and add them to the user's favorites. Up to three favorites are then loaded into the quick access on the device overview page and can be opened using the star.

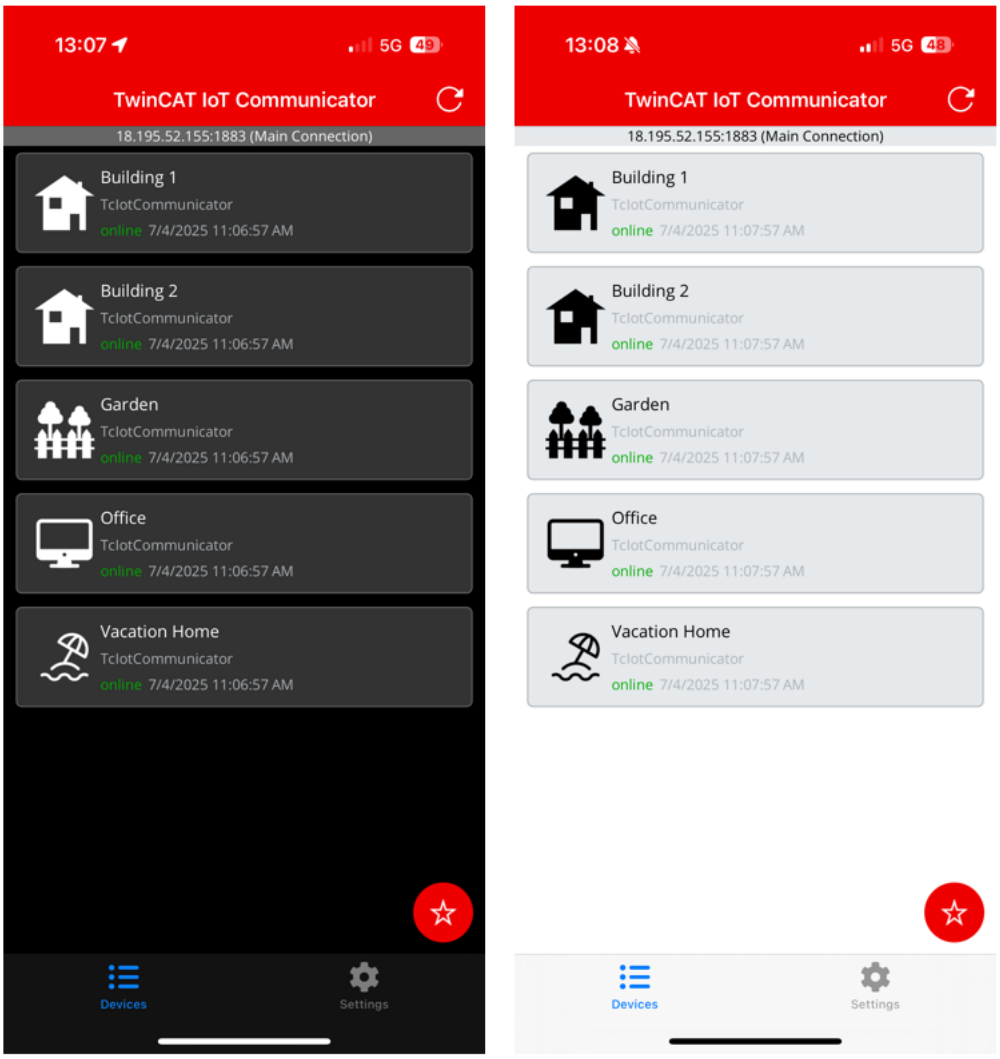


If no favorites have been created, the star is not displayed. It is also possible to establish a connection to the favorites directly via the app icon.

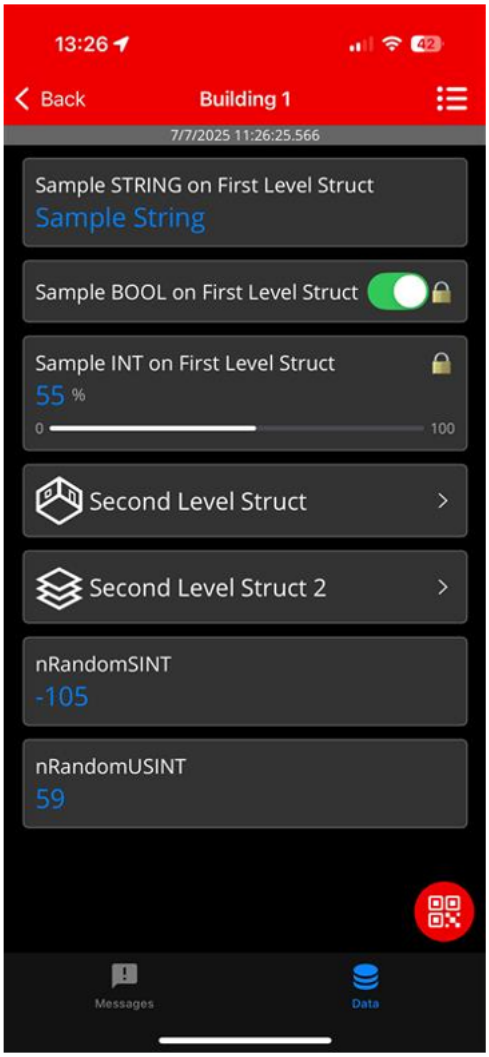


7.2 Device overview

All IoT Communicator function blocks currently connected to the same broker and the same topic are displayed in the device overview. The overview of variables for a device can be opened by clicking this device.

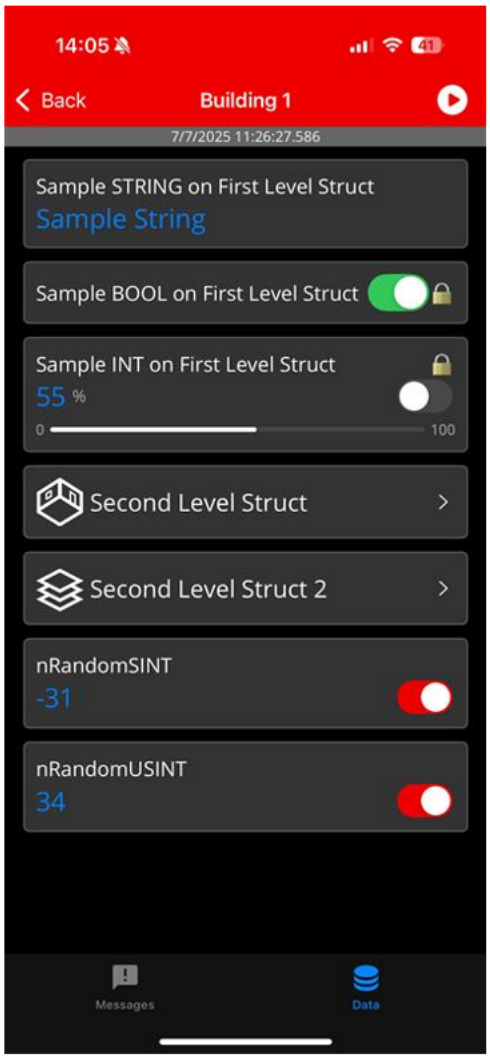


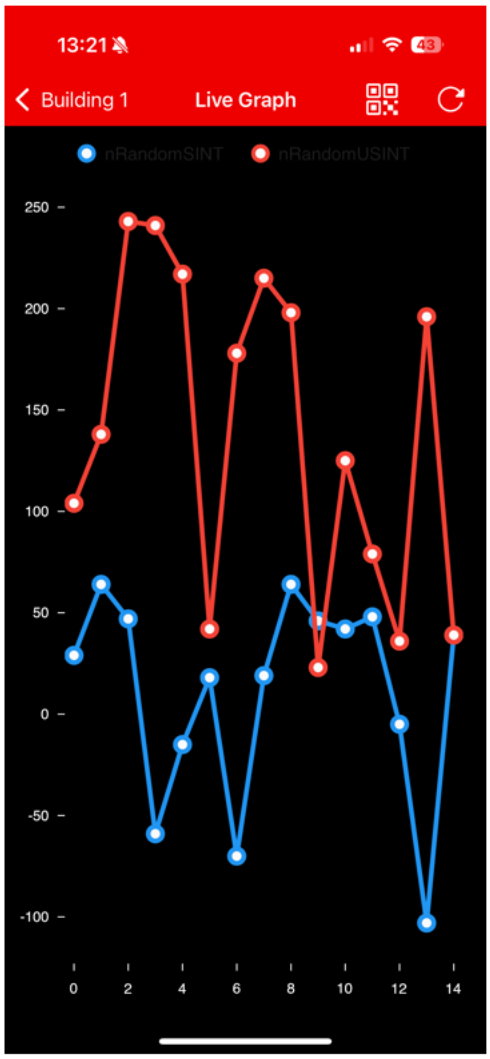
There may be one or more levels, depending on the setup of the structure sent from the PLC.



The graph display is configured via the list symbol in the top right corner. As soon as the user has clicked on the list icon, a selection option appears for the variables that can be displayed in a graph.

After selecting the variables, press the play icon in the top right corner. The graph is then displayed according to the settings.


14:05
Back
Building 1
7/7/2025 11:26:27.586
Sample STRING on First Level Struct
Sample String
Sample BOOL on First Level Struct
Sample INT on First Level Struct
55 %
0
100
Second Level Struct
Second Level Struct 2
nRandomSINT
-31
nRandomUSINT
34
Messages
Data


13:21
Building 1
Live Graph
nRandomSINT
nRandomUSINT
250
200
150
100
50
0
-50
-100
0
2
4
6
8
10
12
14

8 Samples

The samples are divided into two sections. First, there is the Application sample [▶ 77], which builds and explains a sample step by step. On the other hand, there are two samples on Github that can be downloaded and used directly as PLC code.

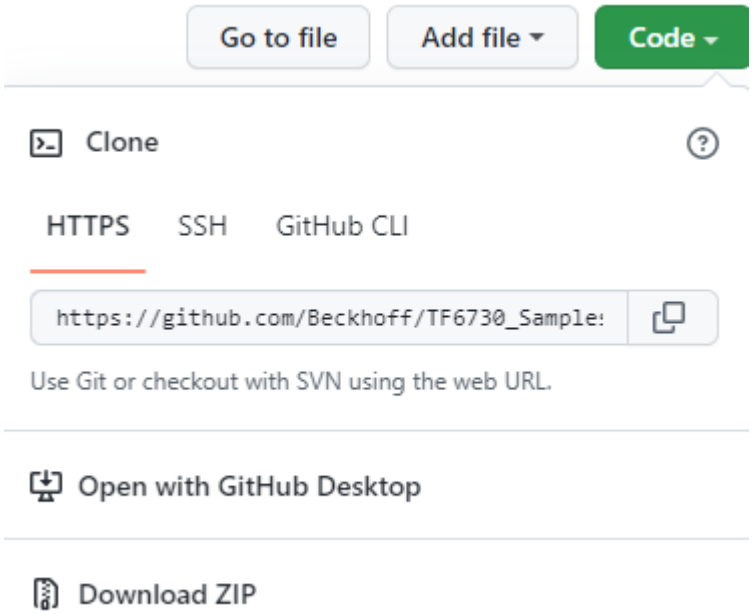
The first sample (TF6730_FullSample) covers the standard functions of the app, while the second sample (TF6730_Widget-Sample) deals with the special extensions for building automation.

Overview

Sample	Description
TF6730_FullSample	Sample of the standard app functionalities.
TF6730_WidgetSample	Sample of widget extensions for building automation.

Downloads

The sample code for the projects described in the overview can be obtained from the corresponding repository on GitHub: https://github.com/Beckhoff/TF6730_Samples. There you have the option to clone the repository or download a ZIP file containing the sample.



8.1 Application sample

Creation of the PLC program

Defining a structure

Create a structure within which you define the process data to be sent. Assign attributes for the declared variables to define their representation in the app (see Attributes [▶ 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
```

Configuration

In the main program, declare an instance of the function block FB_IotCommunicator. Define the outputs according to your connection data (see FB_IotCommunicator [▶ 52]). In addition, declare the structure with the process data to be sent and an instance of the TON timer function block.

```
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
```

Establishing a connection

Cyclically call the Execute method in the implementation part of the main program via the instance of the function block FB_IotCommunicator to maintain the connection to the broker and thus enable sending and receiving of data and messages (see Execute [▶ 55]).

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

Sending data

Send the process data to the broker with a sample rate of 500 ms. To this end, call the instance of the timer function block with the corresponding input variables and the SendData method of the function block FB_IotCommunicator (see SendData [▶ 55]).

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

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

A nested structure must be transferred here for structuring over several levels. The following structure shows a simple example of how nesting can be continued indefinitely.

```
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
```

Receiving and evaluating commands

Call the function block FB_IotCommand and its methods to receive and evaluate commands (see FB_IotCommand [▶ 59]).

```
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.stLevel1.nDoubleCounter),
SIZEOF(stData.stSecondLevel.stLevel1.nDoubleCounter), E_IotCommunicatorDatatype.type_BOOL);
    END_IF
    fbIoT.fbCommand.Remove();
END_IF
```

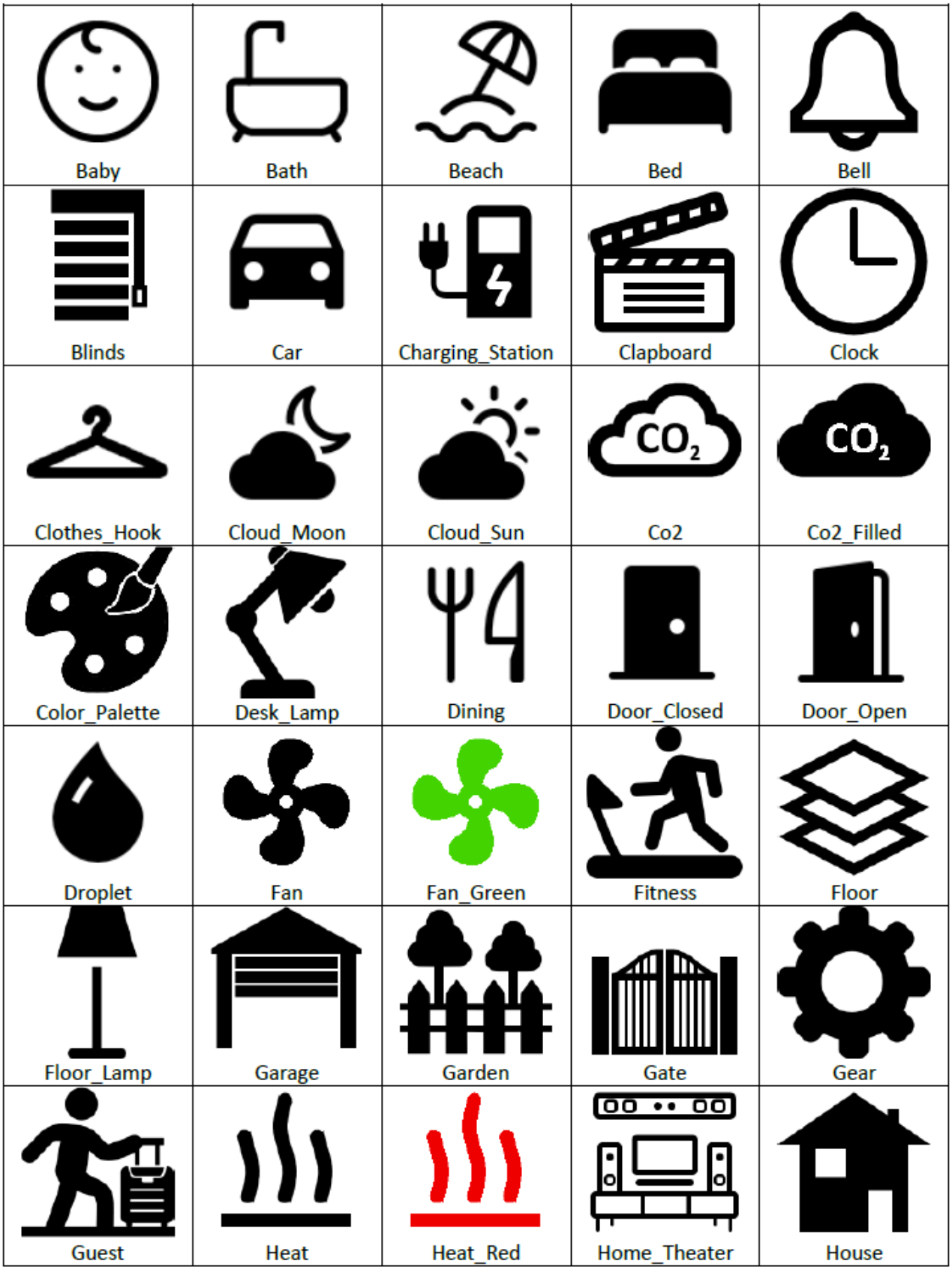
Sending (push) messages

Call the SendMessage method of the function block FB_IotCommunicator to send a (push) message to the broker (see SendMessage [▶ 56]).

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

9 Appendix

9.1 List of available icons



9.2 List of available colors

The available colors are the colors displayed in the Colors Class of Windows (https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.colors). Both the strings and the hexadecimal values can be used. The first two digits of the hexadecimal representation represent the opacity, the other 6 are the representation of the color.

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 and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems

- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Apple and Safari are trademarks of Apple Inc., registered in the U.S. and other countries and regions.

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

Chrome, Chromium and Google are trademarks of Google LLC.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:
www.beckhoff.com/tf6730

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