Manual | EN

TF6730-TF6735

TwinCAT 3 | IoT Communicator (-App)
Table of contents

1 Foreword .............................................................................................................................................. 5
  1.1 Notes on the documentation ........................................................................................................... 5
  1.2 Safety instructions ............................................................................................................................ 6
  1.3 Notes on information security .......................................................................................................... 7

2 Overview ............................................................................................................................................... 8

3 Installation ............................................................................................................................................ 9
  3.1 System requirements .......................................................................................................................... 9
  3.2 Installation ....................................................................................................................................... 9
  3.3 Licensing ........................................................................................................................................ 9

4 Technical introduction .......................................................................................................................... 12
  4.1 MQTT ............................................................................................................................................. 12
    4.1.1 Broker ..................................................................................................................................... 12
    4.1.2 Topic structure .......................................................................................................................... 12
    4.1.3 Quality of Service ...................................................................................................................... 12
  4.2 Security .......................................................................................................................................... 14
    4.2.1 Authentication ............................................................................................................................ 14
    4.2.2 Encryption ................................................................................................................................ 14

5 Configuration ...................................................................................................................................... 15
  5.1 Attributes ...................................................................................................................................... 15
  5.2 Nested structures .............................................................................................................................. 17
  5.3 Limitation of decimal places .......................................................................................................... 17
  5.4 Navigation via QR code .................................................................................................................. 18

6 PLC API .............................................................................................................................................. 21
  6.1 Function blocks ............................................................................................................................... 21
    6.1.1 FB_IotCommunicator .................................................................................................................. 21
    6.1.2 FB_IotCommand ........................................................................................................................... 24
  6.2 Data types ....................................................................................................................................... 25
    6.2.1 ST_IotCommunicatorTls ............................................................................................................ 25
    6.2.2 E_IotCommunicatorDatatype .................................................................................................... 26
    6.2.3 E_IotCommunicatorTlsVersion ................................................................................................ 26

7 App .................................................................................................................................................... 27
  7.1 Settings .......................................................................................................................................... 28
    7.1.1 Connection settings .................................................................................................................... 28
    7.1.2 App settings ............................................................................................................................... 30
  7.2 Device overview .............................................................................................................................. 32

8 Samples .......................................................................................................................................... 35
  8.1 Application sample .......................................................................................................................... 35

9 Appendix .......................................................................................................................................... 37
  9.1 Support and Service ........................................................................................................................ 37
1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared 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 prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
with corresponding applications or registrations in various other countries.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th></th>
<th>DANGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serious risk of injury!</td>
<td></td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk of injury!</td>
<td></td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal injuries!</td>
<td></td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damage to the environment or devices</td>
<td></td>
</tr>
<tr>
<td>Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
<td></td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
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](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](https://www.beckhoff.com/secinfo).
2 Overview

The function blocks of the Tc3_IoT Communicator 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.

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

3.1 System requirements

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

3.2 Installation

No separate setup is required for the TF6730 IoT Communicator function. The required driver components are supplied together with the TwinCAT setup (XAE and XAR).

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.

![Solution Explorer](image)

- 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”).

![Manage Licenses](image)

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

![Order Information](image)
7. Click **7-Day Trial License...** to activate the 7-day trial license.

   ![License Activation Dialog](image)

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

   ![Enter Security Code](image)

   - Please type the following 5 characters: Kg8T4

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 [12])
- Retain messages and Quality of Service 0 & 1 (see Quality of Service [12])

4.1.2 Topic structure

The TC3_IoTCommunicator PLC library sends and receives data via a structure of MQTT topics defined by Beckhoff.

To link a TwinCAT PLC to the MQTT broker, you only have to specify a main topic. See API reference for function block FB_IotCommunicator [21].

4.1.3 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.

**NOTE**

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

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.

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.

NOTE

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**

![Application sample]

**5.2 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**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.17 or higher</td>
<td>App version 1.2.2 or higher</td>
<td>TC3_IotCommunicator</td>
</tr>
</tbody>
</table>

**5.3 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 [30]). The second possibility is to set a certain number of decimal places for a single variable via the PLC attributes (cf. Attributes [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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Decimal Number Precision</th>
<th>Display in the app</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.68678</td>
<td>3</td>
<td>1.687</td>
</tr>
<tr>
<td>1.68678</td>
<td>1</td>
<td>1.7</td>
</tr>
<tr>
<td>1.68678</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.23 or higher</td>
<td>App version 1.2.6 or higher</td>
<td>TC3_IotCommunicator</td>
</tr>
</tbody>
</table>

**5.4 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.
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'; // IP address/hostname of MQTT broker, default 'localhost'
  nPort : UINT := 1883; // (*optional*) Port to use for connection to broker - default: 1883
  sClientId : STRING; // (*optional*) Unique name to connect to MQTT broker
  sMainTopic : STRING; // MQTT topic the TC3 IoT Communicator will communicate with
  sDeviceName : STRING; // e.g. 'Machine XYZ' or 'Room 015 - Kitchen'
  sUser : STRING; // (*optional*) Username for authentication
  sPassword : STRING; // (*optional*) Password for specified User
  stTls : ST_IotCommunicatorTls; // (*optional*) Specify details for secure tls connection
  bRetain : BOOL := TRUE; // (*optional*) default true - if false, data, messages & devices will not be stored
  eQoS : TcIotMqttQos := TcIotMqttQos.AtLeastOnceDelivery; // (*optional*) quality of service between client & broker - default: 1
END_VAR

VAR_OUTPUT
  bError : BOOL;
  hrErrorCode : HRESULT;
  eConnectionState : ETcIotMqttClientState;
  bConnected : BOOL;
  fbCommand : FB_IoTCommand; //provides functionality to receive commands
END_VAR
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sHostName</td>
<td>STRING</td>
<td>sHostName can be specified as the host name or as the IP address. If no information is provided, the local host is used.</td>
</tr>
<tr>
<td>nPort</td>
<td>UINT</td>
<td>The host port is specified here. (Default: 1883)</td>
</tr>
<tr>
<td>sClientId</td>
<td>STRING</td>
<td>The client ID can be specified individually. If no ID is specified, it is generated.</td>
</tr>
<tr>
<td>sMainTopic</td>
<td>STRING</td>
<td>Here you specify the main topic in which the data and messages are sent.</td>
</tr>
<tr>
<td>sDeviceName</td>
<td>STRING</td>
<td>Here you can enter the name of the device to which the data and messages belong.</td>
</tr>
<tr>
<td>sUser</td>
<td>STRING</td>
<td>Optionally, a user name can be specified.</td>
</tr>
<tr>
<td>sPassword</td>
<td>STRING</td>
<td>A password for the user name can be entered here.</td>
</tr>
<tr>
<td>sTLS</td>
<td>STRING</td>
<td>If the broker offers a TLS-secured connection, the required configuration can be implemented here. The parameter structure is of type ST_IotCommunicatorTls [25].</td>
</tr>
<tr>
<td>bRetain</td>
<td>BOOL</td>
<td>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.</td>
</tr>
<tr>
<td>eQoS</td>
<td>TcIotMqttQos</td>
<td>The Quality of Service (QoS for short) can be set with this setting.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>TRUE if an error situation occurs.</td>
</tr>
<tr>
<td>hrErrorCode</td>
<td>HRESULT</td>
<td>Returns an error code if the bError output is set.</td>
</tr>
<tr>
<td>eConnectionState</td>
<td>ETcIotMqttClientState</td>
<td>Indicates the state of the connection between client and broker as enumeration ETcIotMqttClientState.</td>
</tr>
<tr>
<td>bConnected</td>
<td>BOOL</td>
<td>TRUE if there is a connection between the client and the broker.</td>
</tr>
<tr>
<td>fbCommand</td>
<td>FB_IoTCommand</td>
<td>Provides all the necessary functionality to evaluate received data (&quot;Commands&quot;). This output is of type FB_IoTCommand [24].</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute [23]</td>
<td>Method for background communication with the TwinCAT driver. This method must be called cyclically.</td>
</tr>
<tr>
<td>sendData [23]</td>
<td>Method for sending data to the specified MQTT message broker.</td>
</tr>
<tr>
<td>sendMessage [23]</td>
<td>Method for sending a (push) message to the specified MQTT message broker.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.0</td>
<td>IPC or CX (x86, x64, ARM)</td>
<td>Tc3_IotCommunicator</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bConnect</td>
<td>BOOL</td>
<td>The connection to the broker is established when bConnect is set to TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bConnect must remain set to maintain the connection. The connection to the broker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is cut off by calling the Execute() method with FALSE as the input.</td>
</tr>
</tbody>
</table>

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
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pMachineStruct</td>
<td>PVOID</td>
<td>Address for the structure in which the device variables are declared.</td>
</tr>
<tr>
<td>nStructSize</td>
<td>UINT</td>
<td>Size of the structure specified in pMachineStruct.</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendData</td>
<td>BOOL</td>
<td>The method returns TRUE if the call was successful.</td>
</tr>
</tbody>
</table>

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

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMessage</td>
<td>STRING</td>
<td>Text of the (push) message to be sent to the broker.</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendMessage</td>
<td>BOOL</td>
<td>The method returns the return value TRUE if the call was successful.</td>
</tr>
</tbody>
</table>

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.

### 6.1.2 FB_IotCommand

The function block provides functions for evaluating received commands. It must not instantiated, since it is already declared at the output of FB_IotCommunicator [21] 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>TRUE if an error situation occurs..</td>
</tr>
<tr>
<td>hrErrorCode</td>
<td>HRESULT</td>
<td>Returns an error code if the bError output is set.</td>
</tr>
<tr>
<td>bAvailable</td>
<td>BOOL</td>
<td>TRUE, if a new command is available</td>
</tr>
<tr>
<td>sVarName</td>
<td>STRING</td>
<td>If bAvailable is TRUE, sVarName contains the name of the variable that was received.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetValue</td>
<td>Method for accessing the value of the command, if bAvailable is TRUE</td>
</tr>
<tr>
<td>Remove</td>
<td>Method for discarding the currently available command</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.0</td>
<td>IPC or CX (x86, x64, ARM)</td>
<td>Tc3_IotCommunicator</td>
</tr>
</tbody>
</table>

### 6.1.2.1 GetValue

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

**Syntax**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pValue</td>
<td>PVOID</td>
<td>Address of the variable to which the received value is to be written</td>
</tr>
<tr>
<td>nSize</td>
<td>UDINT</td>
<td>Size of the variable specified in pValue</td>
</tr>
<tr>
<td>eDatatype</td>
<td>E_IotCommunicatorDatatype</td>
<td>Data type of the variable specified in pValue, based on enum E_IotCommunicatorDatatype</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetValue</td>
<td>BOOL</td>
<td>The method returns TRUE if the call was successful.</td>
</tr>
</tbody>
</table>

### 6.1.2.2 Remove

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

### 6.2 Data types

#### 6.2.1 ST_IotCommunicatorTls

TLS security settings for the MQTT client.

**Syntax**

```plaintext
TYPE ST_IotCommunicatorTls : STRUCT
   eVersion := E_IotCommunicatorTlsVersion.tlsv1_2; // TLS
```

---

TF6730-TF6735  Version: 1.4  25
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eVersion</td>
<td>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.</td>
<td>TLS version to be used, based on enum E_IotCommunicatorTlsVersion.</td>
</tr>
<tr>
<td>sCA</td>
<td>STRING(255)</td>
<td>Certificate of the certificate authority (CA)</td>
</tr>
<tr>
<td>sCert</td>
<td>STRING(255)</td>
<td>Client certificate that is used for authentication at the broker (optional)</td>
</tr>
<tr>
<td>sKeyFile</td>
<td>STRING(255)</td>
<td>Private key of the client</td>
</tr>
<tr>
<td>sKeyPwd</td>
<td>STRING(255)</td>
<td>Password of the private key, if applicable</td>
</tr>
<tr>
<td>bNoServerCertCheck</td>
<td>BOOL</td>
<td>Disables verification of the server certificate validity</td>
</tr>
</tbody>
</table>

6.2.2 E_IotCommunicatorDatatype

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

6.2.3 E_IotCommunicatorTlsVersion

TYPE E_IotCommunicatorTlsVersion :
{
  tlsv1:=0,
  tlsv1 :=1,
  tlsv1 :=2
} INT;
END_TYPE
7 App

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

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**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker Address</td>
<td>IP address or host name of the message broker.</td>
</tr>
<tr>
<td>Port</td>
<td>The port of the message broker. Usually 1883 (MQTT) or 8883 (MQTT TLS).</td>
</tr>
<tr>
<td>Client Id</td>
<td>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 ID of the mobile device is used.</td>
</tr>
<tr>
<td>Topic</td>
<td>Main topic via which the messages from the associated PLC program are communicated.</td>
</tr>
</tbody>
</table>

**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 area "On my iPhone" with iOS. Further information under [Installation of CA certificates](#).
PKCS12 Certificate (PFX) | Referencing the client certificate as a file. Free file access with Android, only in the area "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.
Allow Untrusted Server Certificate | This setting disables the validation of the server certificate.

Advanced Settings

Setting | Meaning
--- | ---
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.

A QR code looks like this, for example:


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

<table>
<thead>
<tr>
<th>Parameter</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>broker</td>
<td></td>
</tr>
<tr>
<td>port</td>
<td></td>
</tr>
<tr>
<td>clientid</td>
<td></td>
</tr>
<tr>
<td>topic</td>
<td></td>
</tr>
<tr>
<td>user</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.1.1 Installation of CA certificates

For the referencing of CA certificates with the two operating systems supported, Android and iOS, these certificates must first be installed. Short guides for both can be found below. Further information is available in the documentation from both manufacturers.

#### Installation of CA certificates - Android

1. Select the path: Settings/Security/Encryption and login data/Install a certificate/CA certificate.
2. Select a CA certificate.
3. Give the certificate a name (optional).
4. The system reports the successful installation of the CA certificate.

#### Installation of CA certificates - iOS

5. Click the CA certificate file (it must be located outside of the TwinCAT IoT folder). The profile is then loaded.
6. Select the path: Settings/General/Profile.
7. Select the profile of the CA certificate.
8. Press **Install** at the top right.
9. Confirm the installation by entering the code.
10. Read the warning notices at the top right and then press **Install** to confirm.
12. Activate the installed certificate under the item **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.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep Screen on</td>
<td>When activated, the mobile device screen will not turn off while the app is open.</td>
<td>FALSE</td>
</tr>
<tr>
<td>Lock Application</td>
<td>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.)</td>
<td>FALSE</td>
</tr>
<tr>
<td>Send Crash Reports</td>
<td>Emails are sent automatically to the Development Dept. in case of crashes.</td>
<td>Consent is requested here in the case of a new installation.</td>
</tr>
<tr>
<td>Data as Default Page</td>
<td>If activated, the Data tab is opened when opening a device, otherwise the Messages tab is opened.</td>
<td>TRUE</td>
</tr>
<tr>
<td>Enable Generate QR Code Button</td>
<td>A button for generating a QR code is displayed on each page within a Device.</td>
<td>TRUE</td>
</tr>
<tr>
<td>Toggle Booleans</td>
<td>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.</td>
<td>FALSE</td>
</tr>
<tr>
<td>Decimal Number Precision</td>
<td>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.</td>
<td>This value is not set by default.</td>
</tr>
<tr>
<td>Smooth Line</td>
<td>When displaying a live graph, the curve of the graph is shown rounded off if this feature is activated.</td>
<td>FALSE</td>
</tr>
<tr>
<td>Auto Panning</td>
<td>If activated, the graph is trimmed to a certain timespan.</td>
<td>TRUE</td>
</tr>
<tr>
<td>Auto Panning Timespan</td>
<td>The timespan to which a graph is trimmed in the Live View.</td>
<td>5s</td>
</tr>
</tbody>
</table>

### 7.2 Device overview

All IoT Communicators currently connected to the same broker 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. After selecting the variables to be displayed, the play symbol must be pressed, which replaces the list symbol after the latter has been pressed. The graph is then displayed according to the settings.
8 Samples

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]).

```plaintext
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 [21]). In addition, declare the structure with the process data to be sent and an instance of the TON timer function block.

```plaintext
fbIoT : FB_IotCommunicator := (
  sHostName := 'YOUR_MQTT_BROKER', // MQTT Broker Address
  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 [23]).

```plaintext
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 [23]).

```plaintext
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.
Receiving and evaluating commands

Call the function block FB_IotCommand and its methods to receive and evaluate commands (see FB_IotCommand [24]).

IF fbIoT.fbCommand.bAvailable THEN
  IF fbIoT.fbCommand.sVarName = 'bLamp1' THEN
    fbIoT.fbCommand.GetValue(ADR(stData.bLamp1), SIZEOF(stData.bLamp1), E_IotCommunicatorDatatype.type_BOOL);
  ELSIF fbIoT.fbCommand.sVarName = 'stSecondLevel.nDoubleCounter' THEN
    fbIoT.fbCommand.GetValue(ADR(stData.stSecondLevel. nDoubleCounter), 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 [23]).

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

9.1 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.

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 her internet pages: https://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
Fax: +49 5246 963 9157
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
Fax: +49 5246 963 479
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963 0
Fax: +49 5246 963 198
e-mail: info@beckhoff.com
web: https://www.beckhoff.com