

Manual | EN

TF6720

TwinCAT 3 | IoT Data Agent

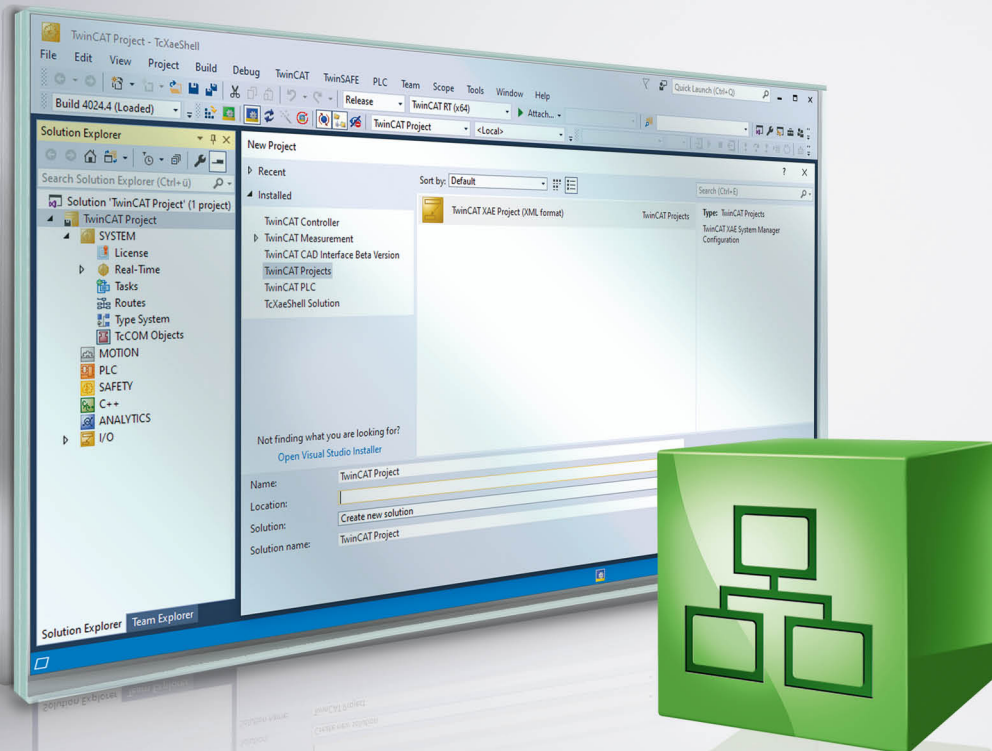


Table of contents

1	Foreword	5
1.1	Notes on the documentation.....	5
1.2	Safety instructions	6
2	Overview	7
3	Installation	8
3.1	System requirements.....	8
3.2	Setup scenarios	8
3.3	Installation	10
3.4	Licensing	13
3.5	Licensing information.....	15
4	Technical introduction	17
4.1	Application examples.....	17
4.1.1	AWS IoT Core.....	17
4.1.2	Bosch IoT Suite	19
4.1.3	Google IoT Core	19
4.1.4	IBM Watson IoT	19
4.1.5	MathWorks ThingSpeak	20
4.1.6	Microsoft Azure IoT Hub	20
4.1.7	Node-RED	21
4.2	Features	22
4.3	Internet connectivity.....	24
4.4	MQTT.....	25
4.5	Startup modes	29
4.6	Security.....	30
4.7	Buffering of data	31
5	Configuration	33
5.1	Quick Start.....	33
5.2	Configurator.....	35
5.2.1	Topology view.....	37
5.2.2	Tree view	37
5.2.3	Mappings	38
5.2.4	Target Browser	38
5.2.5	Cascading Editor	39
5.2.6	Parameter Editor.....	39
5.2.7	Settings.....	41
5.2.8	Error logging	47
5.3	System tray.....	48
5.4	Support Information Report	48
6	Samples	50
6.1	Quick Start.....	50
6.2	Southbound	52
6.2.1	Connect to third-party devices via OPC UA.....	52
6.2.2	Connect to a BC9191 via ADS	54

6.2.3	Connect to a TwinCAT I/O Task via ADS	56
6.2.4	Connect to a TwinCAT TcCOM module via ADS	58
6.3	Northbound	59
6.3.1	Publish data to AWS IoT Core	59
6.3.2	Publish data to AWS Greengrass	61
6.3.3	Publish data to IBM Watson IoT	63
6.3.4	Publish data to Microsoft Azure IoT Hub	65
6.3.5	Publish data to MQTT Message Broker	67
6.3.6	Publish data to TwinCAT IoT Communicator App	69
6.3.7	Subscribe to MQTT Message Broker	70
6.3.8	Use Microsoft Azure IoT Hub Device Twin	72
7	Appendix	74
7.1	Error logging	74
7.2	Error diagnosis	74
7.3	Support and Service	76

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®, 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:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT®

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!

DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.



Tip or pointer

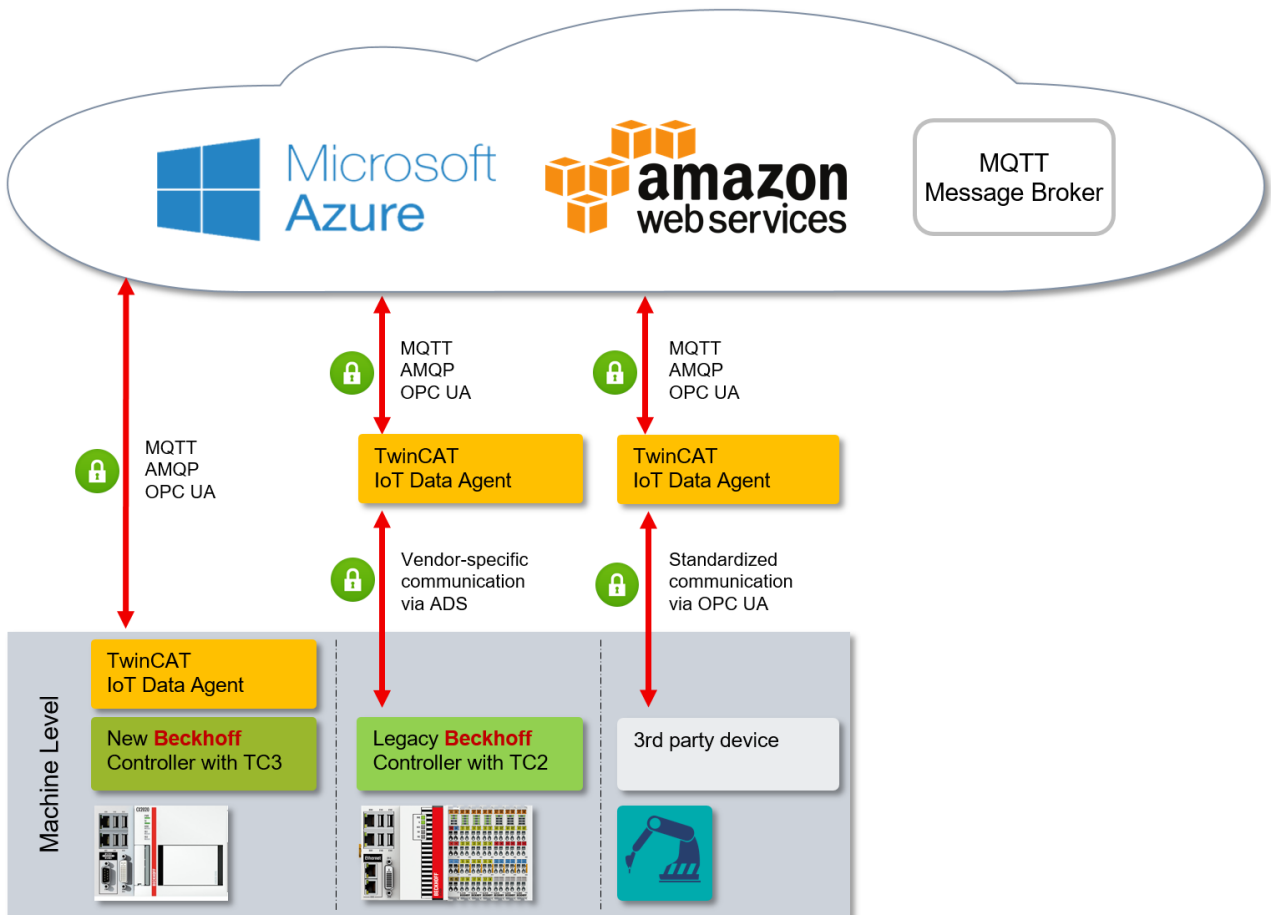
This symbol indicates information that contributes to better understanding.

2 Overview

The TwinCAT Function TC3 IoT Data Agent provides bi-directional connectivity to different cloud services. It is a gateway application that can be installed either on the controller or on a gateway computer. The TC3 IoT Data Agent can be configured to connect different data sources with each other, e.g. a TwinCAT 3 PLC (ADS) and a MQTT Message Broker. In addition, public cloud services can be used, for example AWS IoT and Microsoft Azure IoT Hub. TC3 IoT Data Agent includes an integrated OPC UA client, which allows to connect third-party devices to the cloud. Legacy applications running TwinCAT 2 can be connected by running TC3 IoT Data Agent on a gateway computer and by using ADS or OPC UA connectivity to the TwinCAT 2 system.

When sending or receiving data, users can select from a variety of different data formats, ranging from efficient binary formats to ASCII-based JSON formats. This allows the use of the TC3 IoT Data Agent in different scenarios, e.g. to connect devices to TwinCAT Analytics.

Different communication patterns can be configured to optimize traffic congestion, e.g. polling or on-change patterns.



Components

TC3 IoT Data Agent consists of the following components:

- Core application: background service that provides the logic
- Configurator: graphical tool to create/edit and deploy a configuration for the core application

TwinCAT 3 IoT Data Agent can be installed and used in different setup scenarios, e.g. directly on the controller or on a gateway computer (see [Setup scenarios](#) [▶ 8]).

3 Installation

3.1 System requirements

The following table lists the system requirements for TC3 IoT Data Agent.

Technical data	Description
Operating system	Windows 7/10, Windows Embedded Standard 7
Target platform	PC architecture (x86, x64)
.NET Framework	4.5.2 (only required for configurator)
TwinCAT version ¹	TwinCAT 2, TwinCAT 3
TwinCAT installation level ^{2,3}	TwinCAT 3 XAE, XAR, ADS (build 4022.22 and above)
Required TwinCAT license	TF6720 TC3 IoT Data Agent (includes 4 gates)
Additional licenses ⁴	TF6721 TC3 IoT Data Agent Gate Pack 4 TF6722 TC3 IoT Data Agent Gate Pack 16 TF6723 TC3 IoT Data Agent Gate Pack 64 TF6724 TC3 IoT Data Agent Gate Pack 256

¹ Version of the TwinCAT runtime to which TC3 IoT Data Agent can connect (via ADS, alternatively also via additional functions, e.g. TwinCAT OPC UA)

² A minimum setup level of TwinCAT 3 ADS is required for licensing purposes.

³ XAE is currently required for the configurator.

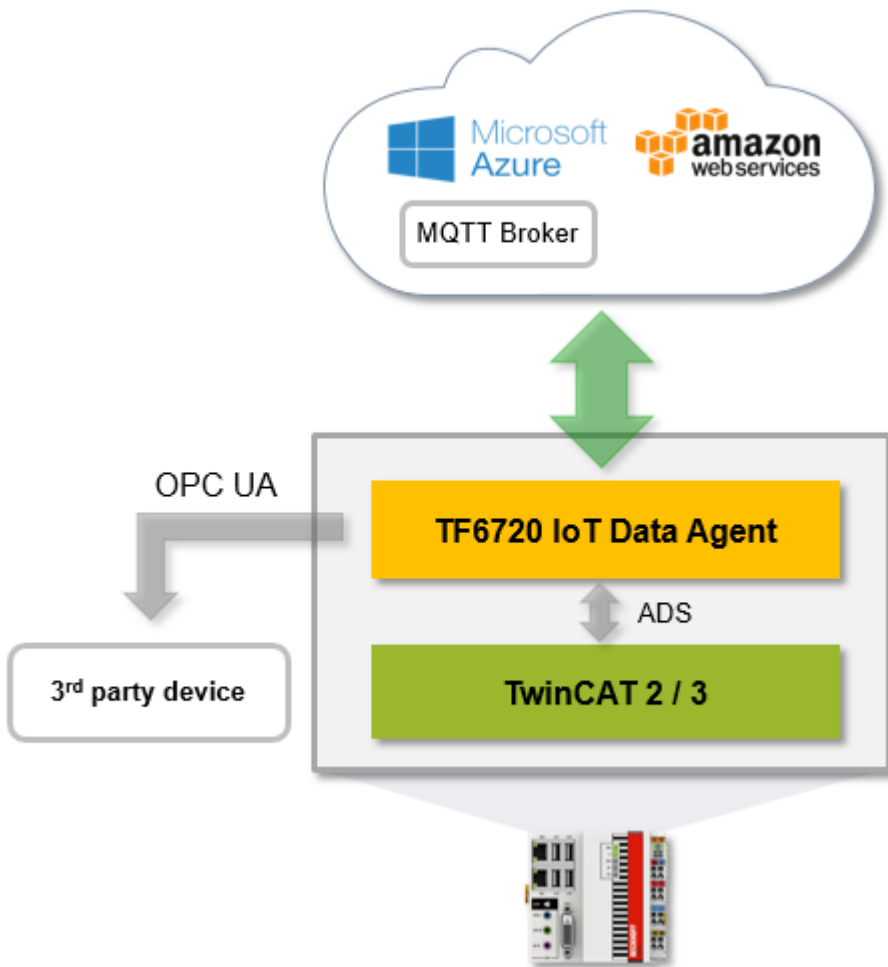
⁴ Every additional license can be acquired once per system and adds to the total amount of gates.

3.2 Setup scenarios

TC3 IoT Data Agent can be used in almost every deployment scenario. It can either be installed and used directly on the industrial controller (see [System Requirements \[► 8\]](#)) or on a gateway computer that aggregates multiple devices or connects to legacy applications like TwinCAT 2. Each setup scenario has its advantages and disadvantages, depending on the system environment and project for which the TC3 IoT Data Agent is to be used.

TC3 IoT Data Agent running on the controller

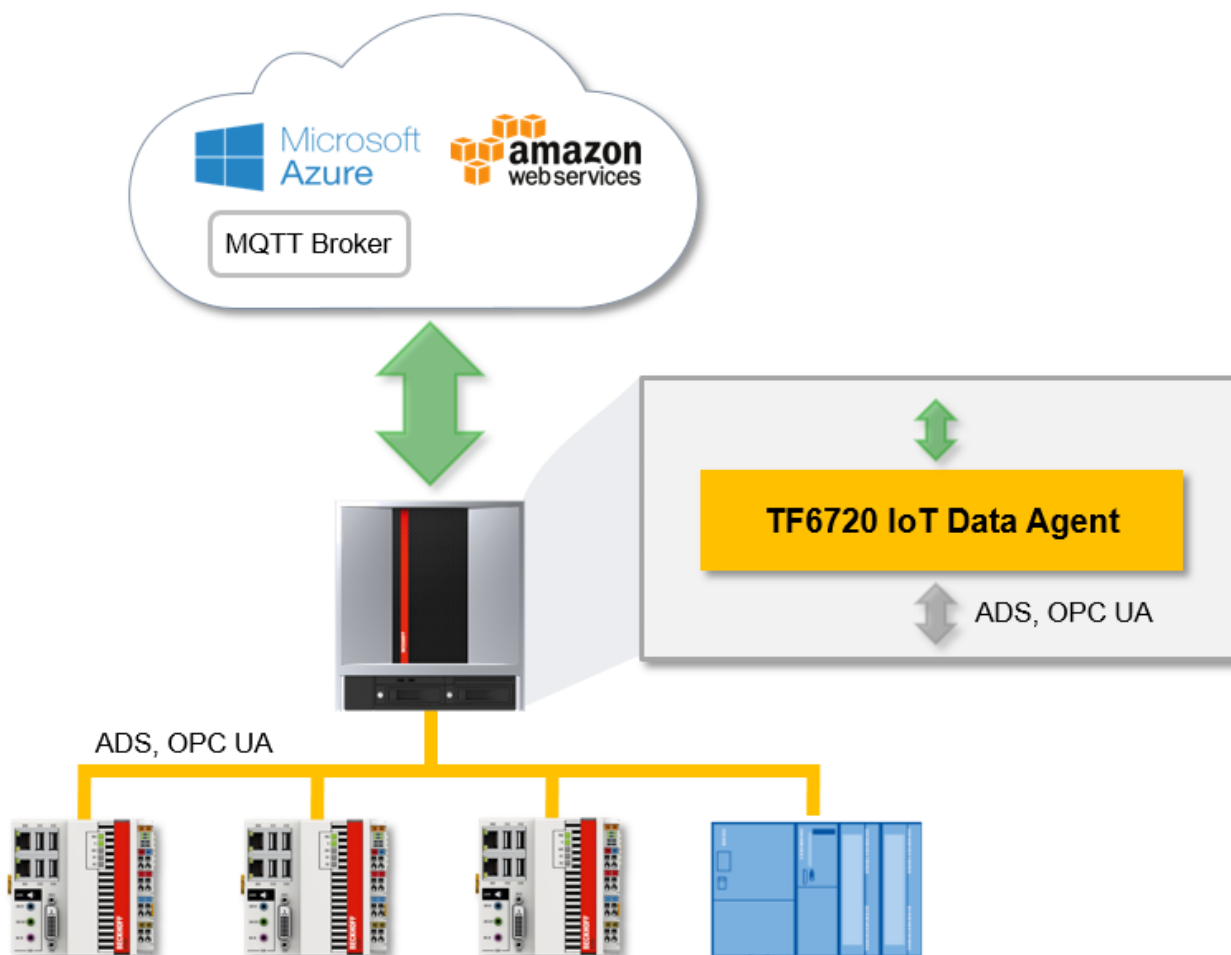
This is the default scenario. TC3 IoT Data Agent is installed directly on a Beckhoff industrial controller (IPC or Embedded-PC) and connects only to the TwinCAT runtime on that system. This is the preferred scenario because every IoT device should have an own connection to the cloud service, although gateway scenarios are possible and sometimes also necessary.



TC3 IoT Data Agent running on a gateway computer

In this scenario, TC3 IoT Data Agent is installed on a gateway computer that aggregates multiple devices, e.g. to connect to existing machine applications in a retrofit scenario. This scenario can also be combined with other TwinCAT Functions, e.g. a TwinCAT OPC UA Server, which is also able to aggregate multiple TwinCAT runtimes. A typical hardware device for such a setup scenario is a Beckhoff C6015 Ultra-Compact Industrial-PC.

Be aware that the network traffic between the TC3 IoT Data Agent and the underlying devices can be very high in this setup scenario because the TC3 IoT Data Agent needs to sample symbols from every connected runtime and this sampling is being done via the network.



TC3 IoT Data Agent configurator

The [TC3 IoT Data Agent configurator](#) [► 35] is a graphical configuration tool. It can be installed on the same computer that executes the TC3 IoT Data Agent core application or on an engineering computer to remotely create/edit and deploy the configuration across the network.

3.3 Installation

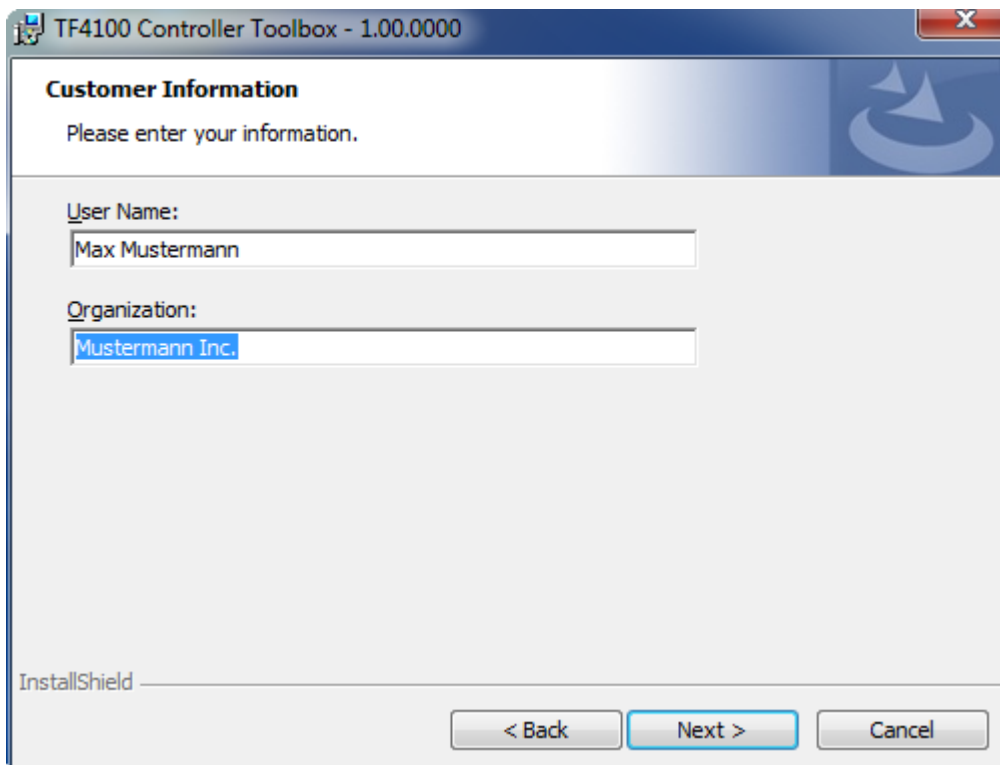
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

- ✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.
- 1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.
 - ⇒ The installation dialog opens.

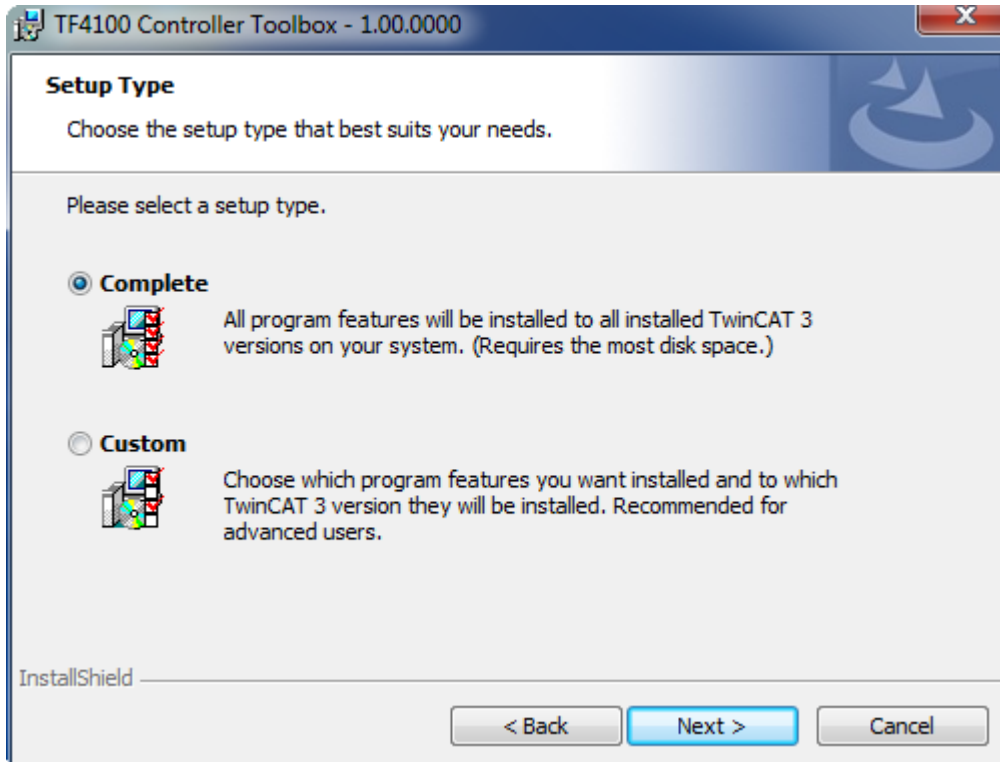
2. Accept the end user licensing agreement and click **Next**.



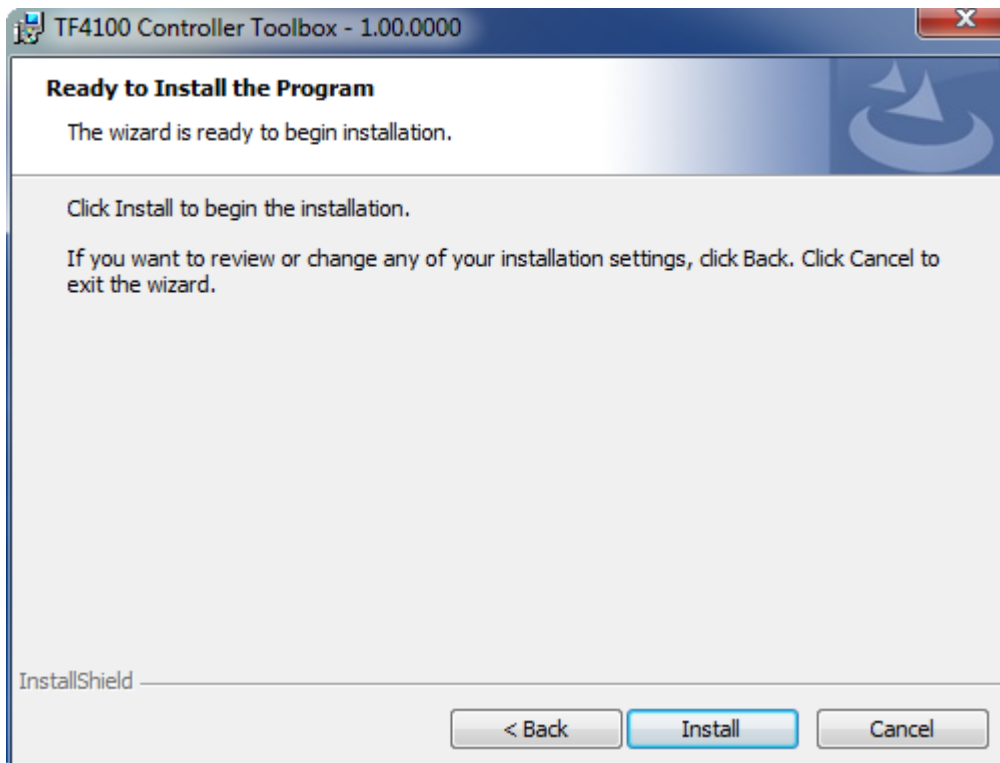
3. Enter your user data.



4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.

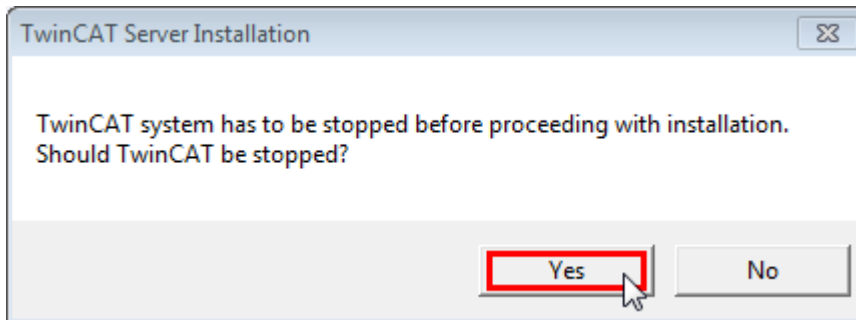


5. Select **Next**, then **Install** to start the installation.

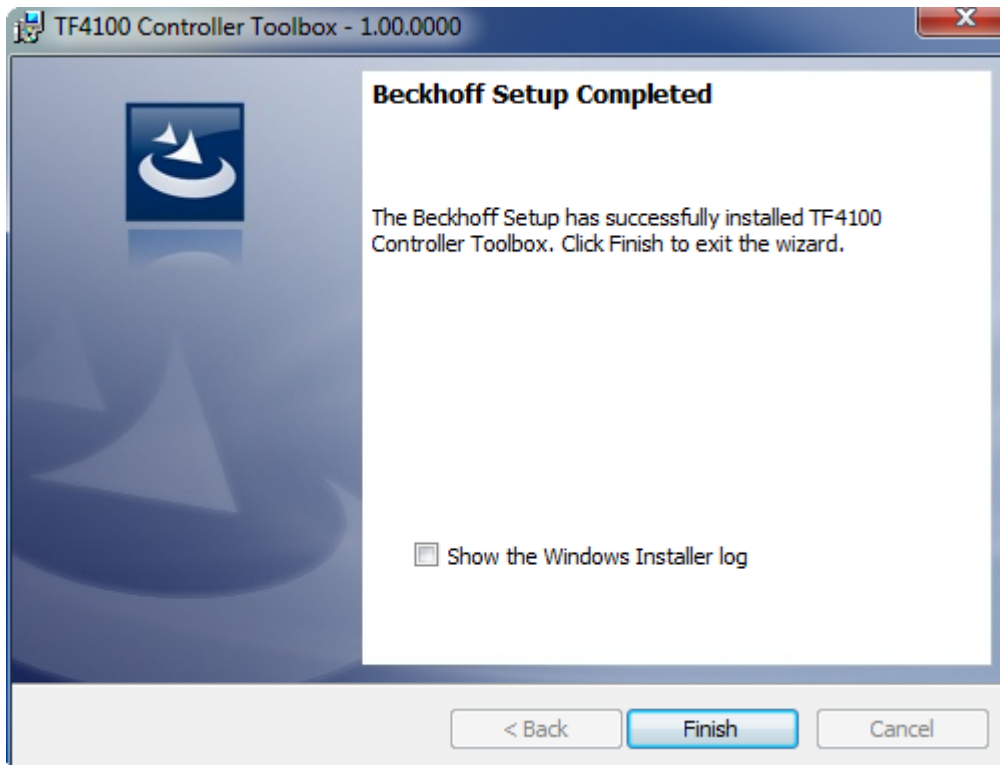


⇒ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.



7. Select **Finish** to exit the setup.



⇒ The TwinCAT 3 Function has been successfully installed and can be licensed (see [Licensing](#) [▶ 13]).

3.4 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](#)".

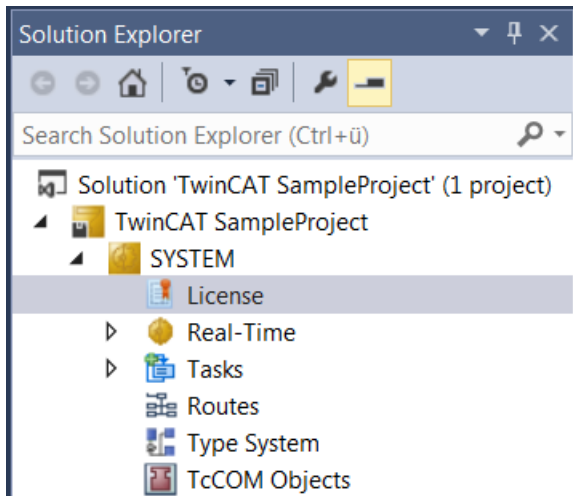
Licensing the 7-day test version of a TwinCAT 3 Function



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

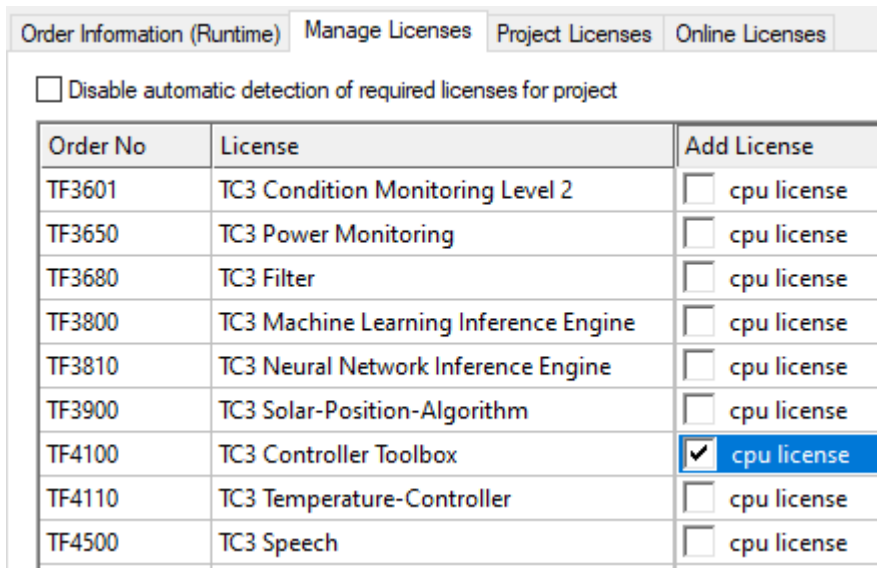
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.

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.



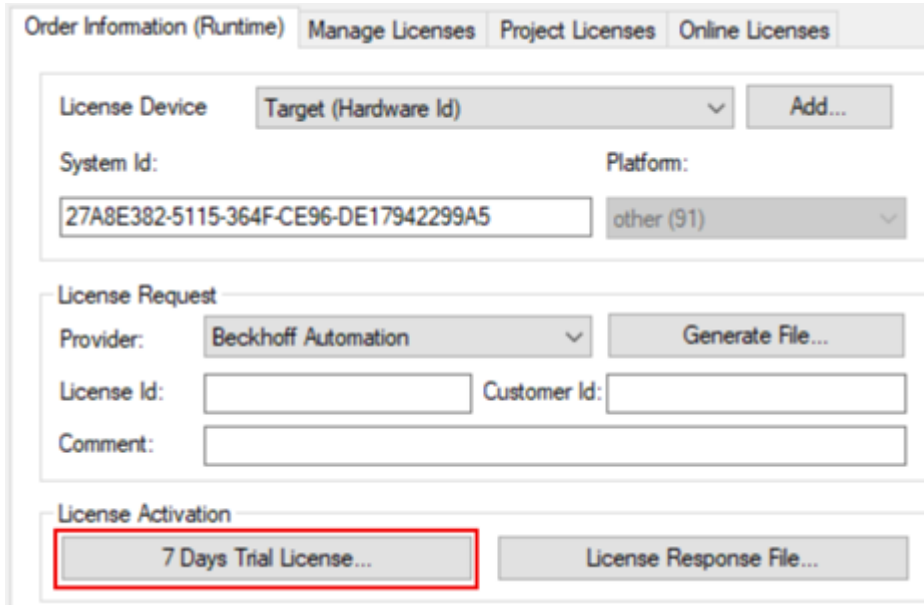
⇒ The TwinCAT 3 license manager opens.

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. "TF6420: TC3 Database Server").

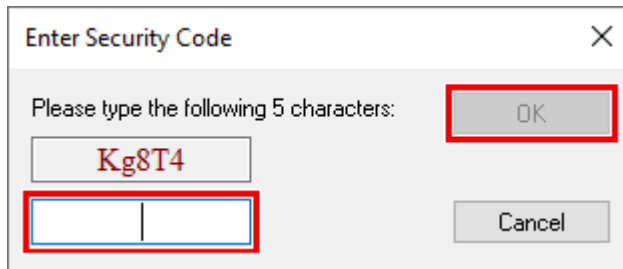


6. Open the **Order Information (Runtime)** tab.
 - ⇒ In the tabular overview of licenses, the previously selected license is displayed with the status "missing".

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.

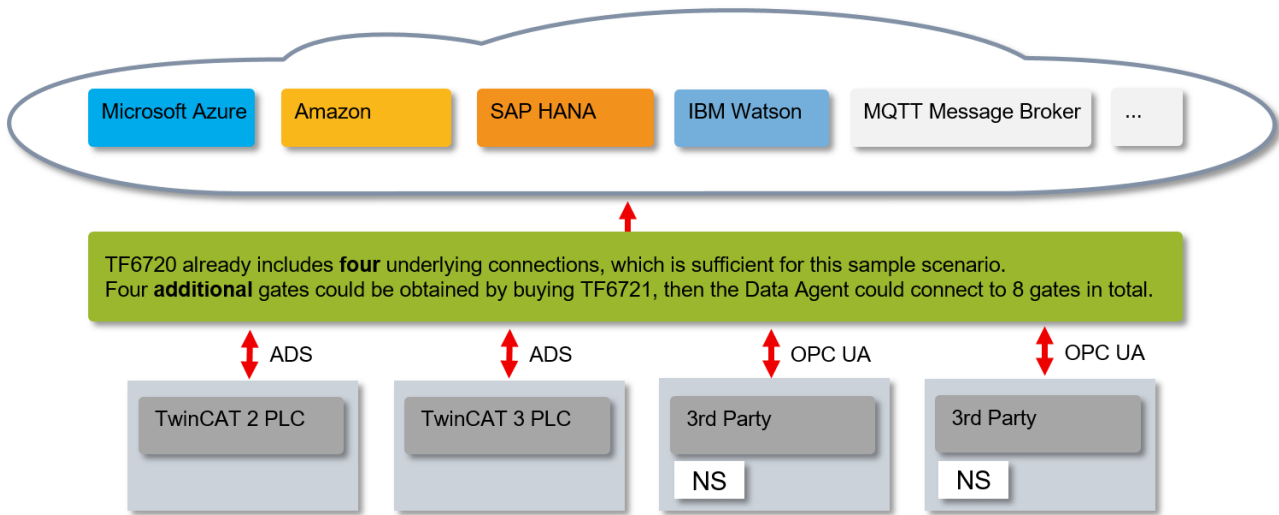
⇒ In the tabular overview of licenses, the license status now indicates the expiry date of the license.

10. Restart the TwinCAT system.

⇒ The 7-day trial version is enabled.

3.5 Licensing information

Licensing of TC3 IoT Data Agent is based on so-called “Gate packs”. Each ADS device or OPC UA server namespace to which the TC3 IoT Data Agent connects is considered to be a “Gate”. The TC3 IoT Data Agent base license “TF6720” already includes an amount of four Gates. Additional Gates can be acquired by buying one of the “Gate pack” licenses (TF6721-TF6724). These licenses are then added to the base license, e.g. TF6720 + TF6721 = 8 Gates.



4 Technical introduction

TC3 IoT Data Agent is a gateway application that connects multiple data sources with each other. A data source is typically a communication protocol, e.g. ADS, OPC UA or MQTT. The TC3 IoT Data Agent is configured via a graphical configurator, which creates or edits an XML-based configuration file. On startup, this file is read by the TC3 IoT Data Agent core application, which is installed as a service and therefore runs in the background of the operating system.

Simply spoken: The XML file tells the TC3 IoT DataAgent what to do.

- Which data sources should be connected
- Which cloud services should be used
- Which symbols (variables) should be exchanged and in which direction
- Which data rates and sampling modes should be used

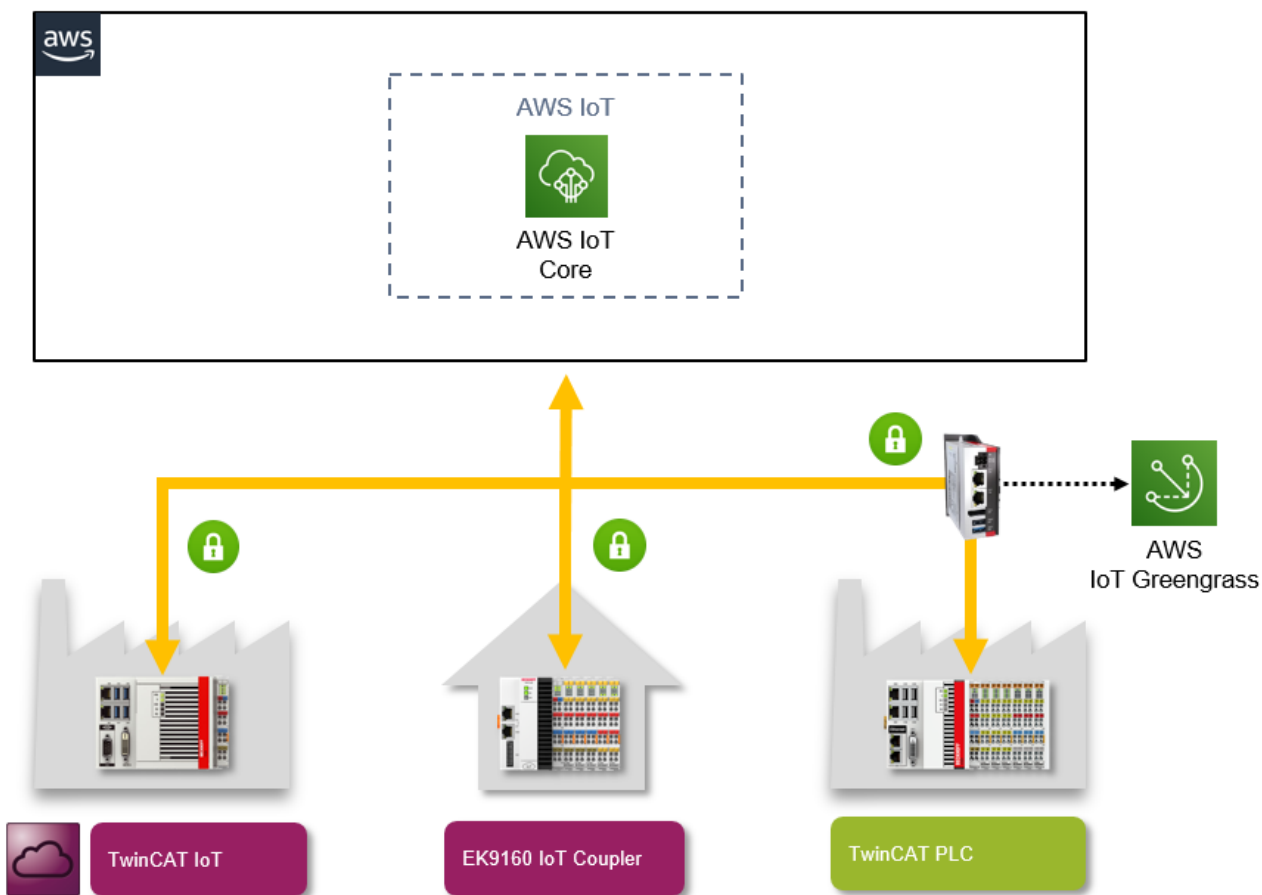
A typical installation of TC3 IoT Data Agent on a controller or gateway computer includes only the TC3 IoT Data Agent core application. The graphical configurator is usually installed on an engineering computer and used to connect to the core application over the network but can also be installed and used on the controller or gateway computer.

4.1 Application examples

The following sections provide some application examples in which TC3 IoT Data Agent can be used.

4.1.1 AWS IoT Core

AWS IoT Core is a managed cloud service that enables connected devices to easily and securely communicate bi-directionally with cloud applications and other devices. Special functionalities, such as the AWS IoT Device Shadow Service, enable communication with devices that are not yet connected.



AWS IoT Core and TwinCAT IoT

AWS IoT Core is based on the MQTT transport protocol. It is therefore possible to use TwinCAT IoT to send or receive messages to or from AWS IoT Core or AWS IoT Greengrass.

Various samples illustrate how to connect to the AWS IoT Core service.

Sample	Product	Description
lotMqttSampleAwsIoT	TF6701	This sample demonstrates how you can use MQTT function blocks within the PLC logic to connect to AWS IoT Core and exchange data.
Data Agent AWS IoT [▶ 59]	TF6720	This sample demonstrates how to configure the TwinCAT IoT Data Agent to connect to AWS IoT Core and exchange data.

In a similar way, the samples can also be applied to AWS IoT Greengrass. The associated AWS IoT Greengrass Core can be installed and operated on a C6015 Industrial PC, for example. Typically, the Greengrass Core (or a Lambda function provided there) initiates data communication with the controller, e.g. via OPC UA. However, since the Greengrass Core has an integrated message broker, this communication direction may be reversed. In this case, TwinCAT IoT would establish a connection to the Greengrass Core via an MQTT channel and exchange data accordingly, for example.

Further Information

Further information about AWS IoT Core can be found in the [AWS IoT Core documentation](#).

4.1.2 Bosch IoT Suite

The Bosch IoT Suite is an IoT platform based on open standards and open source and supports seamless integration into the Bosch IoT eco-system. The Bosch IoT Hub provides devices with various communication interfaces for exchanging data with the Bosch IoT Cloud.

Bosch IoT Suite and TwinCAT IoT

The Bosch IoT Suite includes an MQTT message broker referred to as the Bosch IoT Hub. It is therefore possible to use TwinCAT IoT to send or receive messages to or from the Bosch IoT Suite.

The following sample illustrates how to establish a connection to the Bosch IoT Hub.

Sample	Product	Description
lotMqttSampleBoschIoT	TF6701	This sample demonstrates how you can use MQTT function blocks within the PLC logic to connect to the Bosch IoT Hub and exchange data.

Further Information

i For more information about the Bosch IoT Suite please visit the [official Bosch IoT Suite website](#).

4.1.3 Google IoT Core

Google IoT Core is a managed cloud service that enables distributed devices to exchange data with the Google cloud through a secure transport channel.

Google IoT Core and TwinCAT IoT

Google IoT Core includes an MQTT message broker. It is therefore possible to use TwinCAT IoT to send or receive messages to or from Google IoT Core.

The following sample illustrates how to connect to Google IoT Core.

Sample	Product	Description
lotMqttSampleGoogleIoT	TF6701	This sample demonstrates how you can use MQTT function blocks from within PLC logic to connect to Google IoT Core and exchange data.

4.1.4 IBM Watson IoT

IBM Watson IoT is an IoT suite in the IBM cloud, which offers several services for connecting IoT devices to IBM Bluemix services, processing incoming messages or sending messages to the devices. From a device perspective, the functionalities of IBM Watson IoT enable simple and safe connection of IoT devices with IBM services by facilitating bidirectional communication between the devices and IBM Watson IoT.

IBM Watson IoT and TwinCAT IoT

Since IBM Watson IoT can be reached via the MQTT transport protocol, it is possible to use TwinCAT IoT for sending messages to IBM Watson IoT or receiving messages from it.

Various samples illustrate how to connect to IBM Watson IoT.

Sample	Product	Description
lotMqttSampleIbmWatsonIoT	TF6701	This sample demonstrates how you can use MQTT function blocks to connect to IBM Watson IoT and exchange data.
Publication of data to IBM Watson IoT	TF6720	This sample demonstrates how to configure the TwinCAT IoT Data Agent to establish a connection to IBM Watson IoT and exchange data.

4.1.5 MathWorks ThingSpeak

ThingSpeak™ is an IoT platform from The MathWorks®, well known among other things for the software solutions MATLAB® and Simulink®.

The platform offers (apart from a REST API) an MQTT interface, via which the data from the TwinCAT runtime can be sent to ThingSpeak™. ThingSpeak™ enables the collection, storage, analysis, visualization of and reaction to incoming data. An important point that sets it apart from other platforms is the option to write MATLAB® code in the web browser, which can be used for the analysis and visualization of the data. It is also possible to use existing licenses for toolboxes from the On-premis programming environment in ThingSpeak™.

Functioning

The data ingest and the saving of data take place on the basis of so-called channels. Each channel has 8 fields that can be filled with incoming data. Apart from the 8 data fields there are further meta fields available such as latitude, longitude, altitude or a time stamp. Data published on a channel are stored in a database with the option of a data export (JSON, XML, CSV). The number of messages per time unit that can be sent to a channel depends on the stored ThingSpeak™ license. The MQTT interface is currently based on the sending of strings that are interpreted by the ThingSpeak channel.

Examples of possible actions on ThingSpeak™:

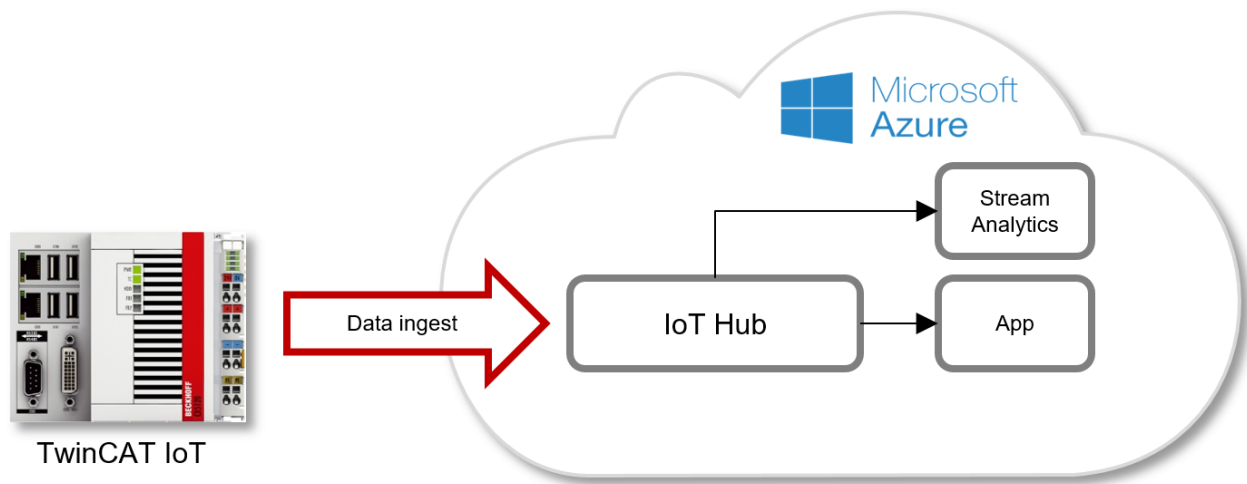
- Send a message to Twitter
- Cyclic execution of Matlab analysis scripts
- Execution of a Matlab analysis script, triggered by channel conditions.

Applications

On account of the data rate – limited by the cloud service – that can currently be sent by the controller to ThingSpeak™, a pronounced edge computing approach is a constructive strategy. MATLAB®/Simulink® models can be integrated in the TwinCAT runtime via the Beckhoff product TE1400 and algorithms for information densification in real time can be executed along with the various TwinCAT functions (Condition Monitoring, filter, etc.). Furthermore, processes with large time constants such as energy data management, building automation, etc. can be handled well with ThingSpeak™.

4.1.6 Microsoft Azure IoT Hub

The Microsoft Azure IoT Hub is an IoT suite in the Azure cloud, which offers several services for connecting IoT devices with Azure services, processing incoming messages or sending messages to the devices. From a device perspective, the functionalities of the Microsoft Azure IoT Hub enable simple and safe connection of IoT devices with Azure services by facilitating bidirectional communication between the devices and the Azure IoT Hub.



Microsoft Azure IoT Hub and TwinCAT IoT

The Microsoft Azure IoT Hub offers several communication interfaces for receiving or sending messages, including MQTT. It is therefore possible to use TwinCAT IoT to send messages to or receive messages from the Microsoft Azure IoT Hub or to control the Device Twin or execute method calls to the device.

Various samples illustrate how to connect to the Microsoft Azure IoT Hub.

Sample	Product	Description
lotMqttSampleAzureIotHub	TF6701	This sample demonstrates how you can use MQTT function blocks within the PLC logic to connect to the Microsoft Azure IoT Hub and exchange data.
<u>Data Agent</u>	TF6720	This sample demonstrates how to configure the TwinCAT IoT Data Agent to establish a connection to the Microsoft Azure IoT Hub and exchange data.
<u>Data Agent Device Twin</u>	TF6720	This sample demonstrates how to configure the TwinCAT IoT Data Agent to establish a connection to the Microsoft Azure IoT Hub and exchange data with the Device Twin.

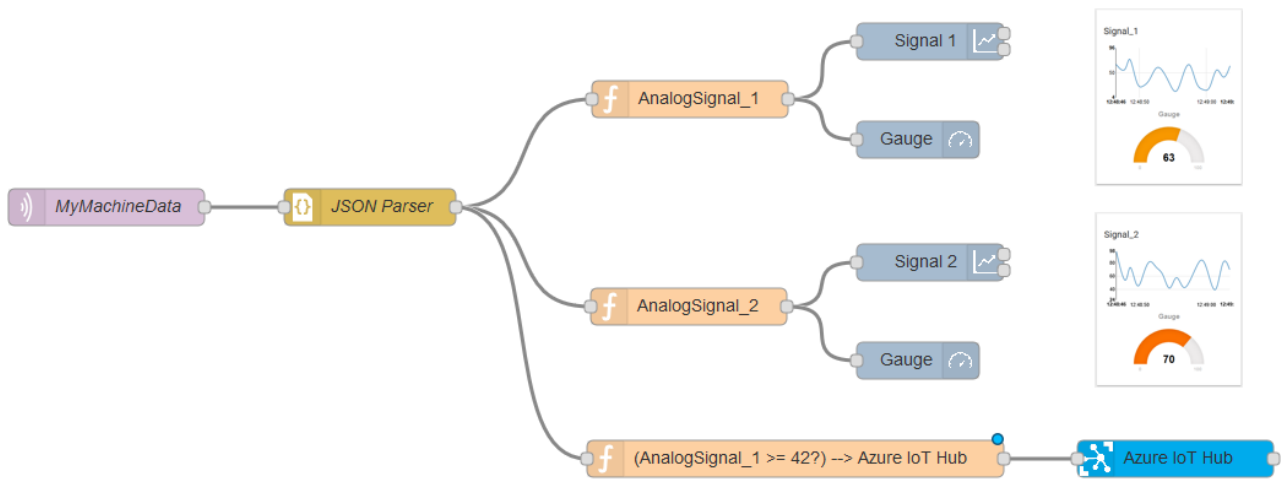
4.1.7 Node-RED

Node-RED is a popular tool that enables workflow-based programming for wiring together hardware devices, APIs and online services in new and interesting ways. It runs on a variety of platforms, e.g. PLC controllers, edge devices, virtual machines or even native services in the cloud. It is a simple open source visual editor for wiring Internet-of-Things services and devices.

The system contains so-called “nodes” that look like icons that can be placed on the canvas and wired together by using drag-and-drop. Each node offers a different functionality. For example, there are simple debug nodes to analyze/check the data flow or MQTT nodes that subscribe to data on a MQTT message broker and insert this data into the workflow. Node-RED can be extended by plugins to enrich its functionalities.

TwinCAT IoT can be used to send machine data to Node-RED and receive commands from the Node-RED workflow. In addition, other TwinCAT functions may be used as well, e.g. the TwinCAT TCP/IP Server.

Node-RED includes built-in support for multiple communication protocols, e.g. MQTT, HTTP, Raw TCP/IP, and so on.



In order to connect TwinCAT IoT via MQTT with a Node-RED workflow, a message broker is required as an intermediate. Another alternative would be to use TwinCAT IoT HTTPS/REST with the built-in HTTPS/REST web server of Node-RED.

Sources

- <https://nodered.org/>
- <https://learn.adafruit.com/raspberry-pi-hosting-node-red/what-is-node-red>

4.2 Features

The following table gives an overview of the TC3 IoT Data Agent features. The features can be categorized as follows:

- General: Basic features of the TC3 IoT Data Agent, e.g. sampling patterns, etc.
- Gates: Supported cloud services and protocols, e.g. ADS, MQTT, OPC UA, etc.
- Data Format: Supported data formats when sending/receiving data to/from a cloud service
- Symbol: Supported data types and additional symbol-related features

Gates

	Description	Version
ADS	ADS devices can be used either via symbolic access or Igrp/Ioff combination. TwinCAT 2/3 I/O channels as well as TwinCAT 2/3 PLC runtimes, TwinCAT 3 TcCOM modules and BC Controllers can be accessed via ADS.	0.7.0.1 and above
OPC UA	The TC3 IoT Data Agent can connect to any OPC UA server that supports the opc.tcp transport channel. In addition, OPC UA certificates and username/password authentication can be used to secure the communication channel to the server.	0.7.0.1 and above
MQTT [▶ 25]	Generic MQTT message brokers can be used to send/receive data. The message broker needs to support the MQTT TCP transport (no web socket connections are supported at the moment). TLS 1.2 can be used to secure the communication channel either via CA certificate, a client certificate or via a PSK.	0.7.0.1 and above
Microsoft Azure IoT Hub	Public cloud service by Microsoft. Can be used to send/receive telemetry data. The communication channel is secured via the primary or secondary key of the IoT Hub device. The device can be created on the Azure IoT Hub configuration website. In addition, the Azure IoT Hub Device Twin service can be used to update the twin document or retrieve desired updates by backend applications.	0.7.0.1 and above 1.3.2.1 and above for Device Twin feature
AWS IoT	Public Cloud service by AWS. Can be used to send/receive data. Configuration is based on specific settings in MQTT gate. TLS 1.2 with a client certificate is mandatory to secure the communication channel. The certificate can be obtained on the AWS IoT configuration website.	0.16.2.2 and above

Data formats

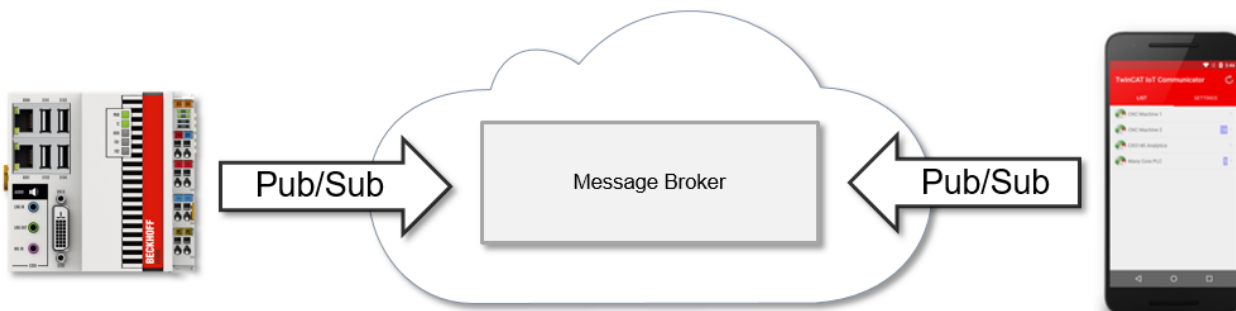
	Description	Version
ADS binary	Binary ADS data format. Can be used for send/receive operations.	0.7.0.1 and above
Simple JSON	“Flat” JSON format which consists of a timestamp and key/value pairs that represent the variables. Can be used for send/receive operations.	0.16.2.2 and above
TwinCAT JSON	Common JSON data format in all TwinCAT IoT and Analytics products as well as the EK9160. See separate documentation article about TwinCAT IoT JSON data format specification. Can be used for send/receive operations.	0.7.0.1 and above
TwinCAT Analytics	Common binary data format in all TwinCAT Analytics products as well as the EK9160.	0.26.2.8 and above

Symbols

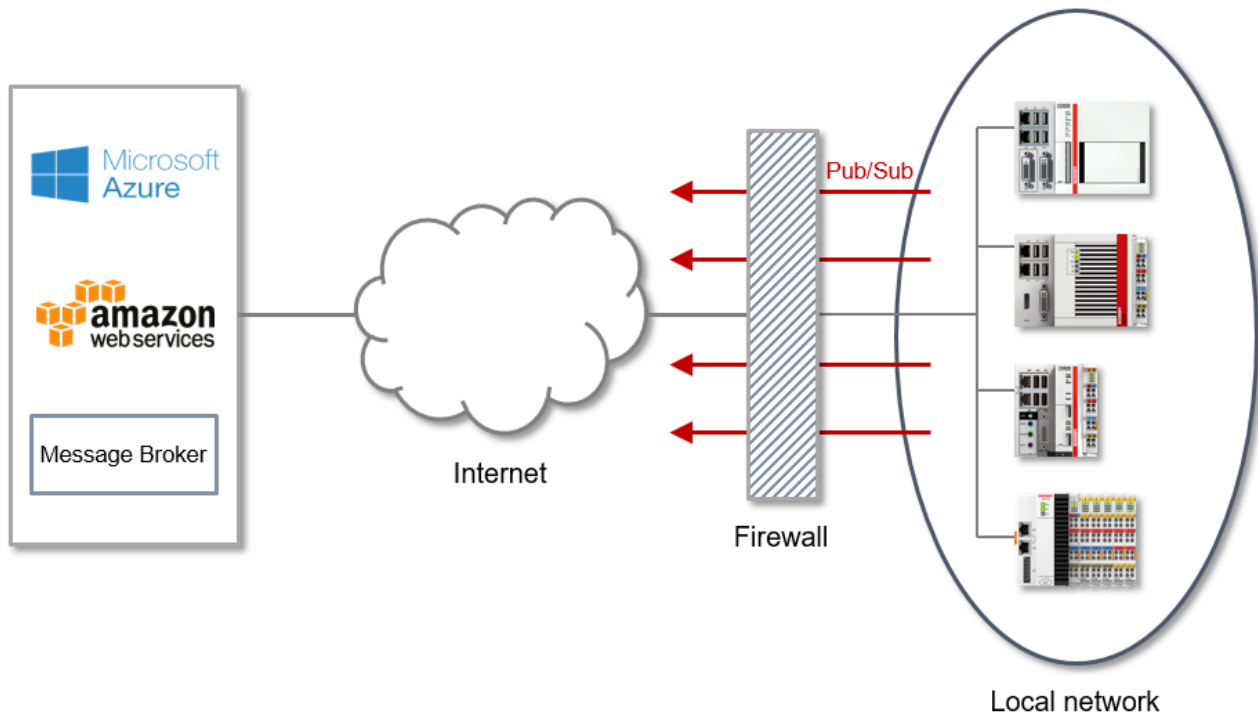
	Description	Version
Simple data types	Simple data types are supported: Int16, Int32, Int64, UInt16, UInt32, Float, Double, ...	0.7.0.1 and above
Strings	Strings are supported	0.7.0.1 and above
Arrays	Arrays with simple and complex data types are supported for publish direction	1.2.17.7 and above
Structured Types	Structured types are supported for publish direction	1.2.17.7 and above
Trigger Symbol	TriggerSymbols can be used to set up a “send on demand” use case	0.8.2.21 and above
Type Conversion	Type conversion can be used to convert between different data types	0.8.2.21 and above
Multi-link	Link one subscriber symbol with multiple publisher symbols. (not the other way around)	0.24.1.4 and above
Units	Units are supported as meta-data on a symbol. This setting depends on the data format and gate.	0.26.2.8 and above
Rounding float types	Rounding of float types to a specified amount of decimal places is supported	1.3.2.1 and above

4.3 Internet connectivity

One of the main advantages of TwinCAT IoT and the EK9160 IoT coupler is that the cloud-related communication protocols are based on publisher/subscriber (“pub/sub”) and broker-based connectivity principles.



From a firewall point-of-view, publisher/subscriber is based on only outgoing data communications. This not only allows an easy integration into the existing IT infrastructure, but is also very convenient for security reasons, as no incoming firewall ports need to be opened (“port forwarding”) to send and receive data. Because of this technical advantage, it is not required or not recommended to place TwinCAT IoT and EK9160 products on the internet edge or untrusted network. Instead, these products can and should be used within local, trusted network boundaries – secured by a firewall from the Internet.



The following table gives an overview about the commonly used communication protocols and their corresponding network ports. Note that this port overview depends on the specific environment and scenario and is based on commonly used default settings.

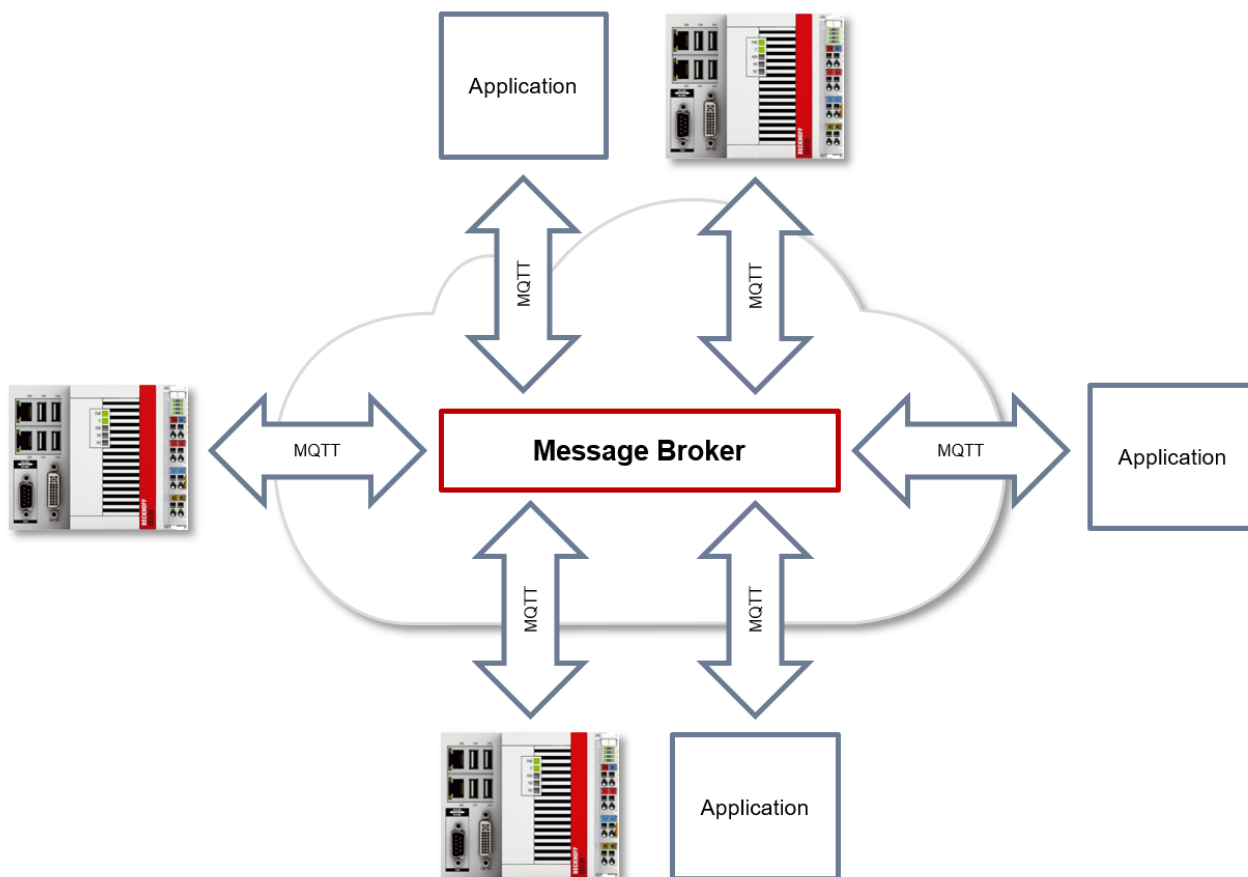
Service	Protocol	Network port	Notes
MQTT Message Broker	MQTT	1883/tcp	
MQTT Message Broker	MQTT with TLS	8883/tcp	
Microsoft Azure IoT Hub	AMQP	5671/tcp	
Microsoft Azure IoT Hub	MQTT with TLS	8883/tcp	
AWS IoT	MQTT with TLS	8883/tcp	
AWS Greengrass	MQTT with TLS	8883/tcp	

Microsoft Azure IoT Hub

i TC3 IoT Data Agent uses AMQP when establishing a connection with the Microsoft Azure IoT Hub.

4.4 MQTT

MQTT(Message Queuing Telemetry Transport) is a publisher/subscriber-based communication protocol, which enables message-based transfer between applications. A central component of this transfer type is the so-called message broker, 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.



Payload

The content of an MQTT message is referred to as payload. Data of any type can be transferred, e.g. text, individual numerical values or a whole information structure.

i Message payload formatting

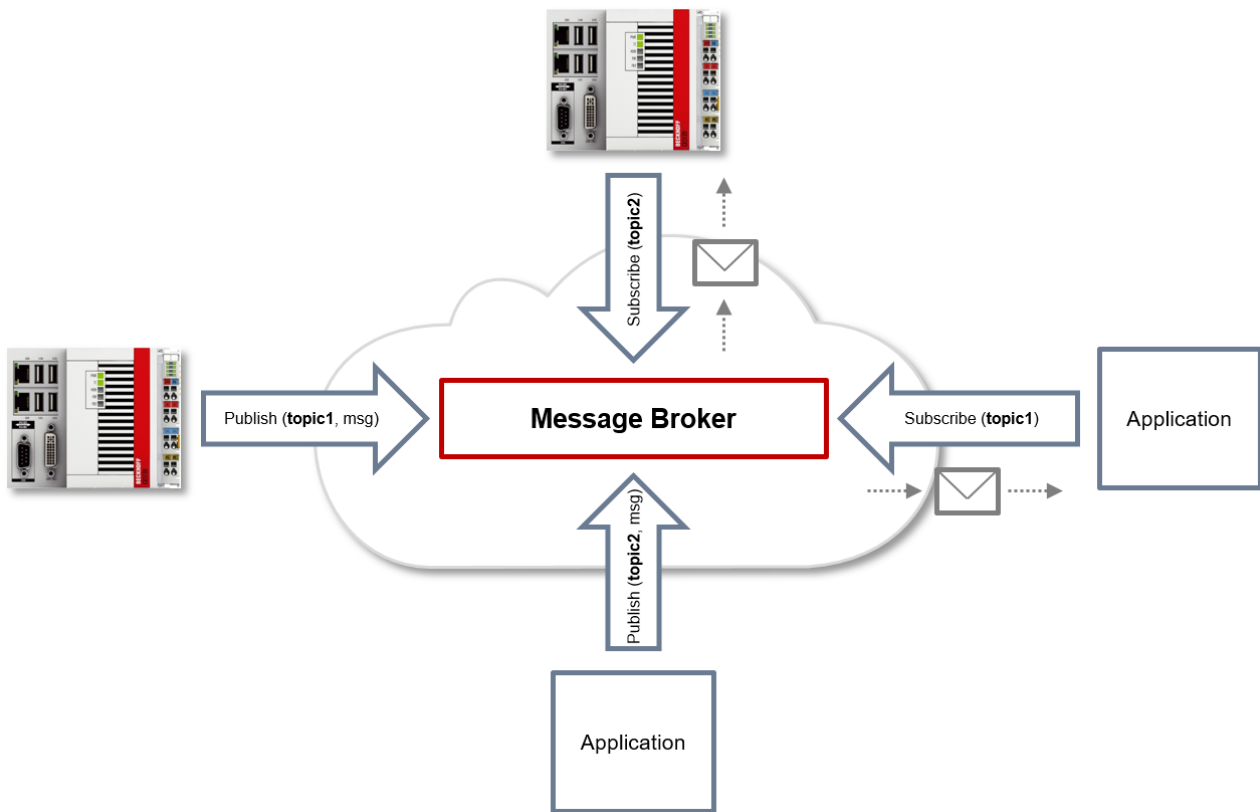
Note that the data type and the formatting of the content must be known to the sender and receiver side, particularly when binary information (alignment) or strings (with or without zero termination) are sent.

Topics

If a message broker is used that is based on the MQTT protocol, sending (publish mode) and subscribing (subscribe mode) of messages is organized with the aid of so-called topics. The message broker filters incoming messages based on these topics for each connected client. A topic may consist of several levels; the individual levels are separated by “/”.

Example: Campus / Building1 / Floor2 / Room3 / Temperature

When a publisher sends a message, it always specifies for which topic it is intended. A subscriber indicates which topic it is interested in. The message broker forwards the message accordingly.



Communication example 1 from the diagram above:

- An application subscribes to “topic1”.
- A controller publishes a message to “topic1”.
- The message broker forwards the message to the application accordingly.

Communication example 2 from the diagram above:

- A controller subscribes to “topic2”.
- An application publishes a message to “topic2”.
- The message broker forwards the message to the controller accordingly.

Wildcards

It is possible to use wildcards in conjunction with topics. A wildcard is used to represent part of the topic. In this case a subscriber may receive messages from several topics. A distinction is made between two types of wildcards:

- Single-level wildcards
- Multi-level wildcards

Example for single-level wildcard:

The + symbol describes a single-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not.

- The receiver subscribes to Campus/Building1/Floor2/+/Temperature
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - NOK

Example for multi-level wildcard:

The # symbol describes a multi-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not. The # symbol must always be the last symbol in a topic string.

- The receiver subscribes to Campus/Building1/Floor2/#
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room1/Humidity - OK

QoS (Quality of Service)

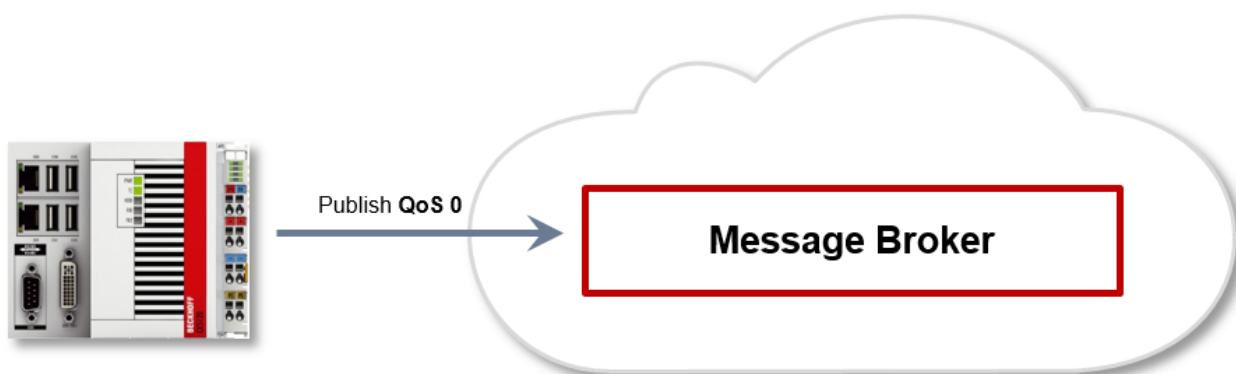
QoS is an arrangement between the sender and receiver of a message with regard to guaranteeing of the message transfer. MQTT features three different levels:

- 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.

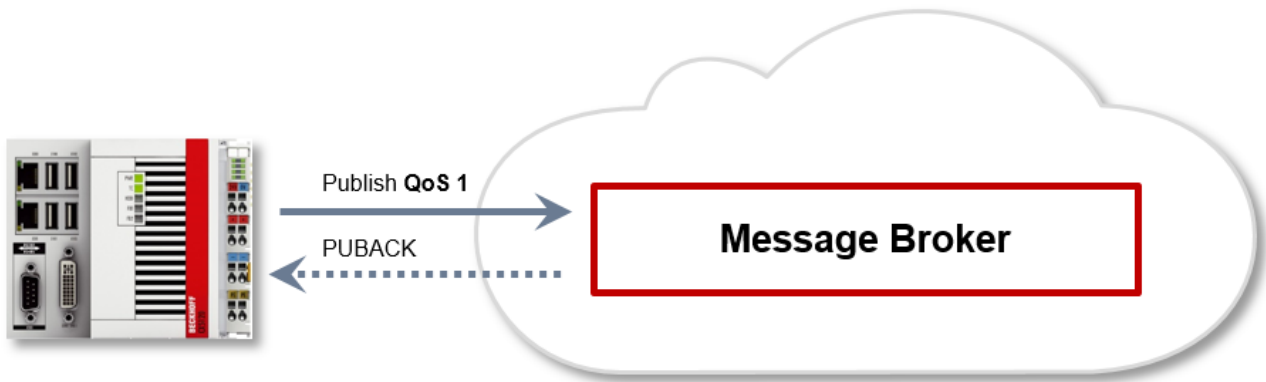
QoS-Level 0

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



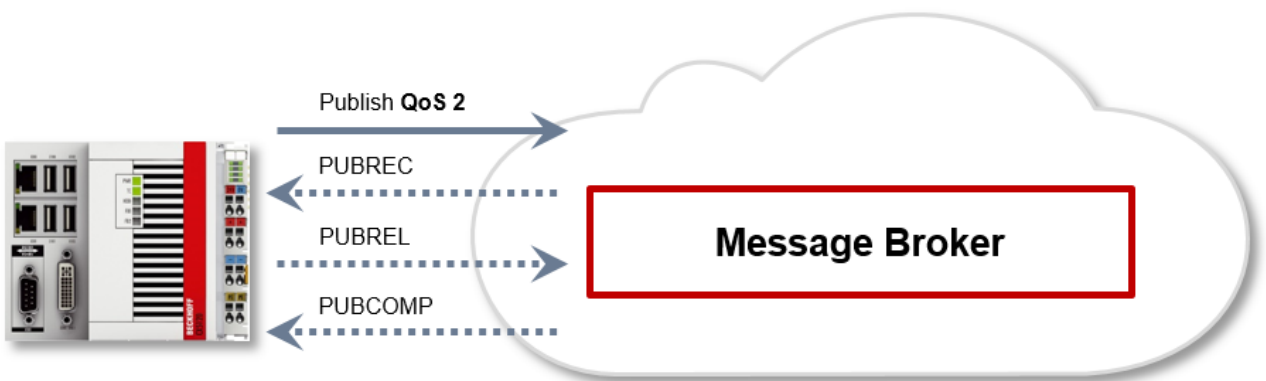
QoS-Level 1

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 acknowledgement 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.



Security

When a connection to the message broker is established, it is possible to use security mechanisms such as TLS, in order to encrypt the communication link or to realize authentication between client and message broker.

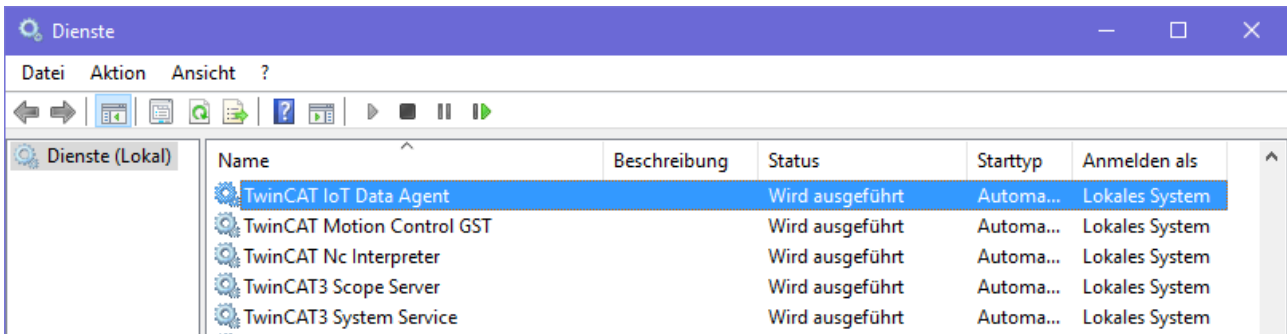
Sources

For further and more detailed information about MQTT we recommend the following blog series:

HiveMq blog: <http://www.hivemq.com/blog/mqtt-essentials/> (the main basis for this article)

4.5 Startup modes

TC3 IoT Data Agent is automatically installed as a Windows service that runs under the local system context (and is therefore independent of the user account logged in) and starts automatically during system boot.



After starting the TC3 IoT Data Agent, the core application waits for a trigger command to start the communication. The trigger command is usually send via the configurator or the system tray application (see [Configurator \[▶ 35\]](#) and [System tray \[▶ 48\]](#)). You can also configure an AutoStart option per configuration file by using the graphical configurator tool.

Command line

As an alternative, the TC3 IoT Data Agent can be executed as a standalone application on the Windows command line. To do so, start the TC3 IoT Data Agent executable file (*TclotDataAgent.exe*) in the installation directory. You can set different startup options by specifying parameters to the process call, e.g. a path to the configuration file or logging settings. Use the parameter “-h” for a list of all startup options.

Examples:

Process call	Description
TclotDataAgent.exe	The configuration file <i>TclotDataAgentConfig.xml</i> from the default config directory %TC_BOOTDIR% is used. Waits for a trigger command to start the communication.
TclotDataAgent.exe -a	Starts the communication after startup automatically. Takes the config file automatically from %TC_BOOTDIR%.
TclotDataAgent.exe -c <i>PathToConfig</i>	Specifies the configuration file to be used. Waits for a trigger command to start the communication.
TclotDataAgent.exe -a -c <i>PathToConfig</i>	Specifies the configuration file to be used. Starts the communication automatically without waiting for a trigger command.
TclotDataAgent.exe -l VERBOSE	The configuration file is retrieved from the TwinCAT boot directory and logging is activated in verbose mode. The Windows command line is used as log output.
TclotDataAgent.exe -l VERBOSE -o <i>PathToFile</i>	The configuration file is retrieved from the TwinCAT boot directory and logging is activated in verbose mode. A file is used as log output.

4.6 Security

Depending on the functionality of each gate, TC3 IoT Data Agent supports several security mechanisms to secure data communication, for example the encryption of messages and client/server authentication.

The following table gives an overview of the different security mechanisms in each supported Gate.

Gate	Security mechanisms	Description
ADS	none	ADS communication should be treated as none secure.
OPC UA	Client/Server authentication Username/password authentication Authorization Data encryption	Depending on the functionalities of the OPC UA server that the TC3 IoT Data Agent should connect to, the following security mechanisms may can be used: <ul style="list-style-type: none"> • Client/Server authentication can be used via X.509 certificates (public/private key) and establishing a trust relationship between the communication participants. • Username/password authentication can be used by an OPC UA client to connect to an OPC UA server. • Authorization can be used on the server to define access levels on OPC UA nodes to configure which user identity may access which nodes on an OPC UA server. • Data encryption can be used to encrypt OPC UA messages on the wire.
MQTT	Client/Server authentication Username/password authentication Authorization Data encryption	Depending on the functionalities of MQTT message broker that the TC3 IoT Data Agent should connect to, the following security mechanisms may be used. <ul style="list-style-type: none"> • Client/Server authentication can be used via X.509 certificates (public/private key) and establishing a trust relationship between the communication participants (client and message broker) • Username/password authentication can be used by a client to connect to the message broker • Authorization can be used on the message broker to define access levels on topics in order to configure which user identity may access which topic on the message broker • Data encryption can be used to encrypt messages on the wire. The TC3 IoT Data Agent uses TLS version 1.2 for securing the communication channel.
Microsoft Azure IoT Hub	Device authentication Data encryption	Every Azure IoT Hub client needs to be registered as a device on the IoT Hub instance. During device registration, a DeviceId and SharedAccessKey is generated, which need to be used during connection establishment for authenticating the device to the IoT Hub instance. In addition, messages are encrypted on the wire.
AWS IoT	Device authentication Authorization Data encryption	Every AWS IoT client needs to be registered as a "thing". A thing/device is configured with X.509 certificates that authenticates the device at AWS IoT and can also be linked with security policies to authorize the device to perform certain actions. In addition, messages are encrypted on the wire. TLS version 1.2 is used to secure the communication channel.

4.7 Buffering of data

The Data Agent offers a buffer function for short-term network failures. It is very important to note that buffering takes place exclusively in RAM. Accordingly, an eye should always be kept on the RAM with this buffer function. If a connection to the target has been restored, the Data Agent begins to process the buffer.

Another point to note is the restrictions of a message brokers to which data are sent. There are message brokers that are limited to a certain number of messages per time unit. Care must be taken when conceiving a Data Agent application that this upper limit is not exceeded when processing a buffer.

Configuration in the Data Agent

A Publisher channel offers the possibility to set the size of a buffer. This size is related to the number of messages to be buffered.

For this reason there is a setting option `InhibitTime` on this Publisher channel. This specifies the time interval with which the messages are to be collected from the buffer as soon as the connection to the target has been restored. In the case of a cyclic Publisher channel the `InhibitTime` **must** be smaller than the `CycleTime`, as otherwise the buffer can never be emptied. In the case of an `OnChange` or `Trigger` channel, the `InhibitTime` indicates the cycle time with which the data are collected from the buffer and published to the Message Broker. If the `InhibitTime` is set to 0, the messages are sent to the target by the Data Agent as quickly as possible.

If new messages arrive during the processing of the buffer, they are appended to the end of the buffer. This ensures the correct order of the data.

5 Configuration

5.1 Quick Start

This section describes how to get started quickly with the TC3 IoT Data Agent.

The communication scenario is a typical use case for the Data Agent: Variables from a southbound device (TwinCAT PLC runtime) are sampled and published to a northbound service (MQTT Message Broker). In this example, the message broker is running on the same computer as the PLC runtime.

To set up this scenario, the following steps are required. All steps are performed on the same computer.

- [Install the software \[► 33\]](#)
- [Prepare the PLC program \[► 33\]](#)
- [Configure the TC3 IoT Data Agent \[► 33\]](#)
- [Display the published messages \[► 35\]](#)

Install the software

1. Install TF6720 IoT Data Agent as described in the [Installation \[► 10\]](#) section.
2. Install Mosquitto MQTT Message Broker with its default setup settings. The Message Broker can be installed anywhere and is connected to via its hostname or IP address.
3. Open a Windows Console, navigate to the installation directory of the Mosquitto MQTT Message Broker (typically *C:\Program Files (x86)\mosquitto*) and start the Broker in verbose mode by executing the following command: `mosquitto.exe -v`

● Mosquitto MQTT Message Broker

I The Mosquitto MQTT Message Broker can be downloaded from www.mosquitto.org. Make sure to follow any installation instructions provided by the Mosquitto installer.

● Mosquitto startup

I By default, the Mosquitto MQTT Message Broker is installed as a Windows service. You can also start the `mosquitto.exe` from a command line and specify parameters like `"-v"` for verbose output. If you want to start the broker from a command line, make sure that the Mosquitto Windows service is not running.

Prepare the PLC program

Set up a TwinCAT 3 PLC project and activate trial licenses for TC1200 (PLC) and TF6720 (TC3 IoT Data Agent). In this sample the PLC program should include the following variables:

```
PROGRAM MAIN
VAR
    nCounter1 : INT;
    nCounter2 : LREAL;
END_VAR
```

These two variables should be incremented cyclically. Add the following implementation code for the PLC program:

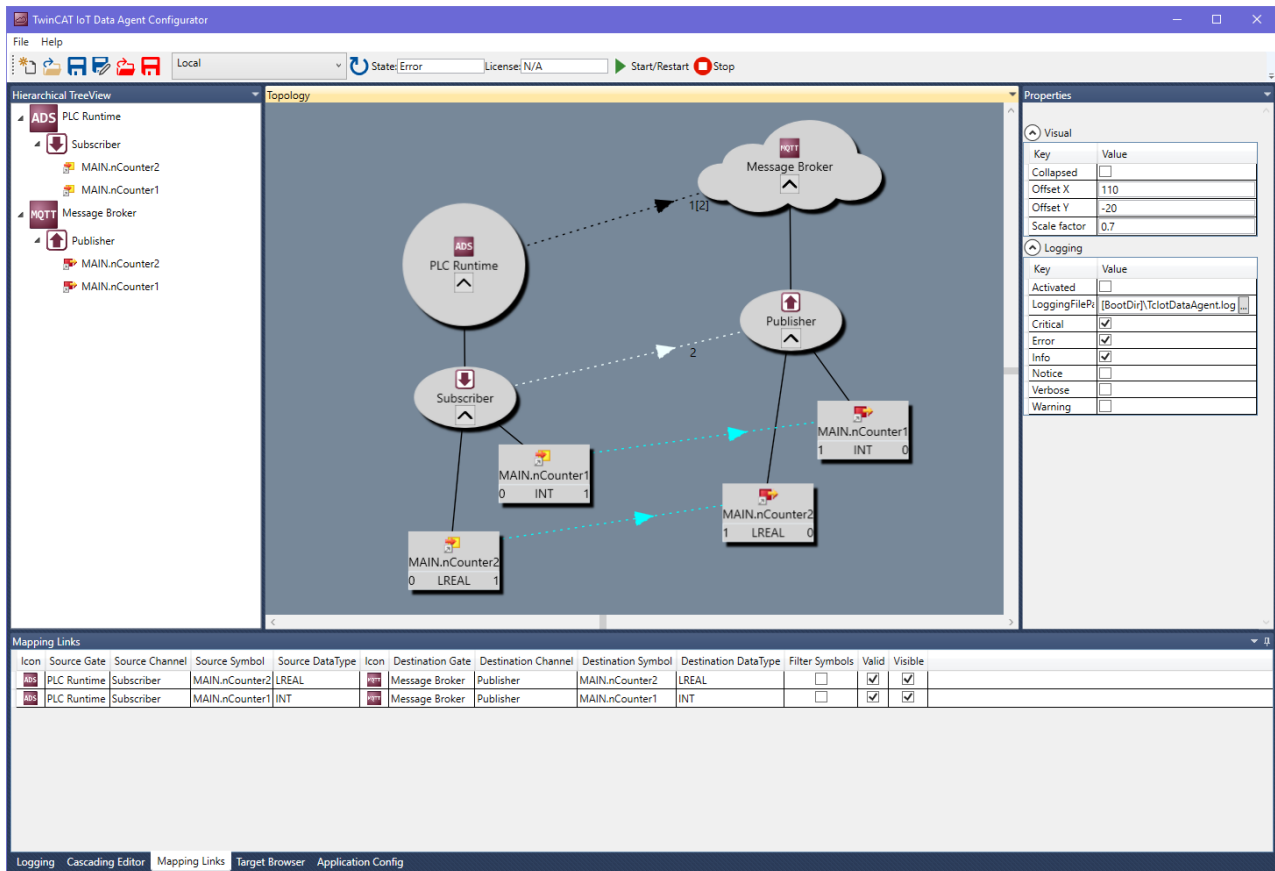
```
nCounter1 := nCounter1 + 1;
nCounter2 := nCounter2 + 0.1;
```

Activate the project and start TwinCAT into run mode. Make sure that the PLC program is running by logging in.

Configure TwinCAT IoT Data Agent

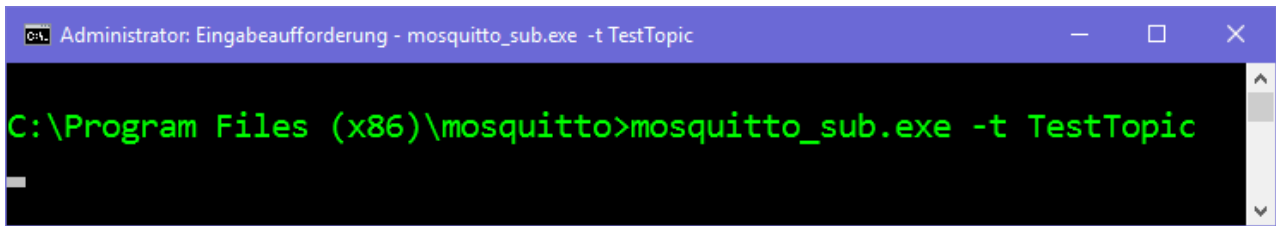
1. Start the TwinCAT IoT Data Agent configurator, either via its link on the Windows start menu or via the system tray application (see [Configurator \[► 35\]](#)).

2. Follow the step-by-step instructions below to create a Data Agent configuration. As an alternative you can also open the configuration file "Sample_QuickStart.xml", which can be found in the Data Agent installation directory, and directly move on with step (m).
3. Start a new configuration by clicking on the **New** icon in the configurator toolbar.
4. In the topology view, right-click a blank area and select **Add Gate (ADS)**.
5. Select the new gate and make sure that the local AmsNetId and ADS port 851 are selected in the properties window
6. In the topology view, right-click the new gate and select **Add Channel (Subscriber)**.
7. Select the new channel and make sure that sampling mode "cyclic" and cycle time "1000" are selected in the properties window.
8. Open the target browser and navigate to your local PLC runtime (port 851).
9. Select the variables "nCounter1" and "nCounter2" and drag them to the subscriber channel in the topology view.
10. In the topology view, right-click a blank area and select **Add Gate (MQTT)**.
11. Select the new gate and make sure that the URL is set to 127.0.0.1 and Port 1883 is being used.
12. In the topology view, right-click the new gate and select **Add Channel (Publisher)**.
13. Select the new channel and make sure that the sampling mode is set to "cyclic". Set the cycle time to "1000" and the formatter to "TwinCAT JSON". The MQTT topic can be set to a topic of your choice, e.g. "TestTopic".
14. Hold the CTRL key and drag the subscriber channel to the publisher channel in order to create a mapping between the source and destination symbols. Note that the destination symbols are automatically created.
15. Activate the configuration by clicking on the **Activate Configuration** button in the toolbar.

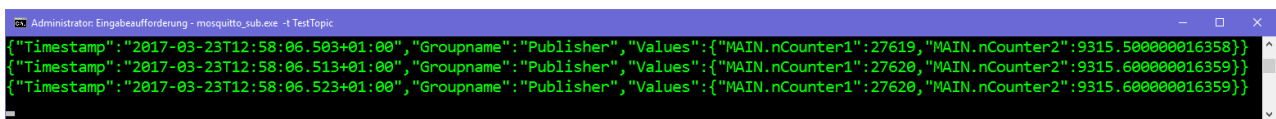


Display the published messages

Open a command line window and navigate to the installation directory of the Mosquitto MQTT Message Broker (typically "C:\Program Files\mosquitto" or "C:\Program Files (x86)\mosquitto"). Start the Mosquitto Subscriber tool by executing the following command:
 mosquitto_sub.exe -t TestTopic.



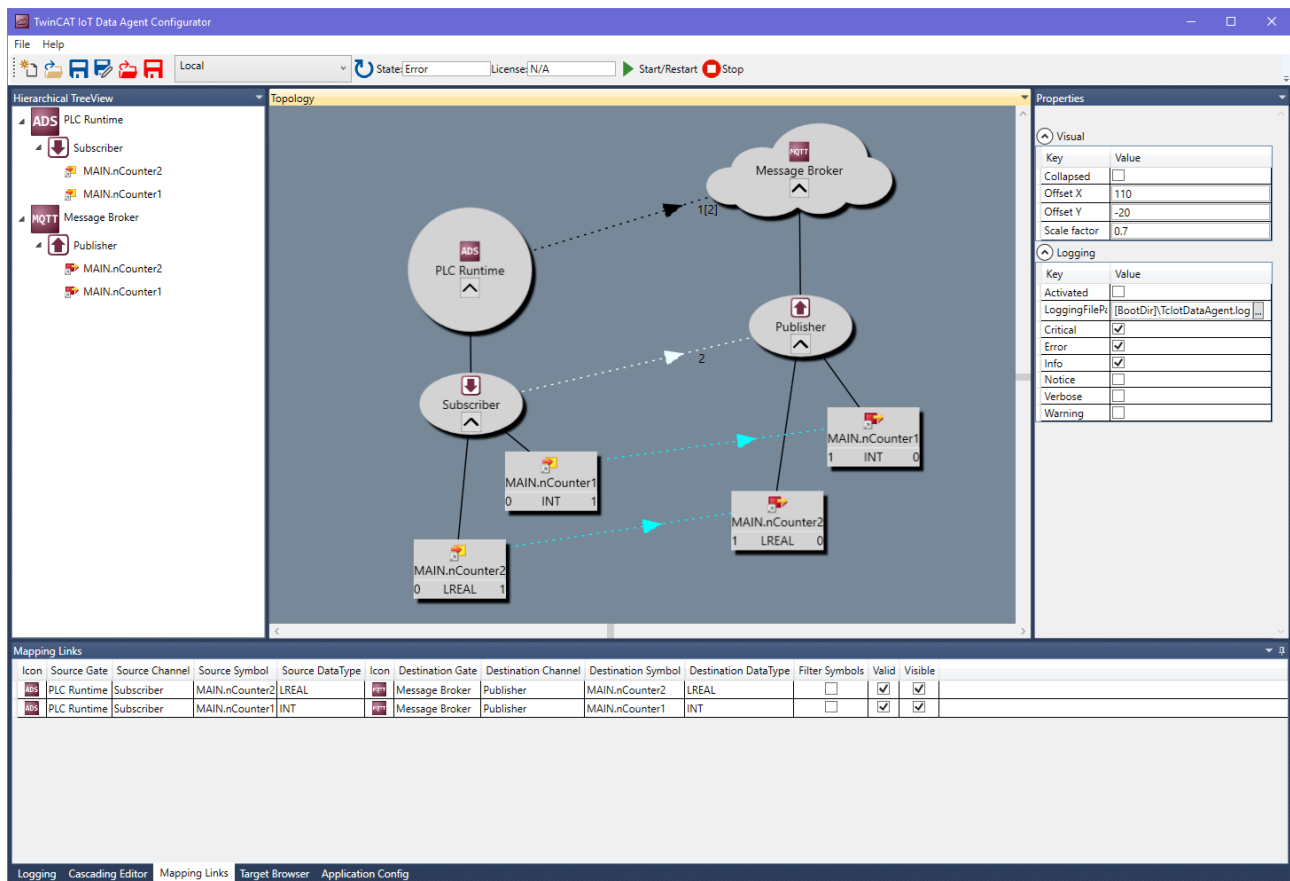
The Mosquitto Subscriber tool will now receive the messages that are published from the TwinCAT IoT Data Agent.



5.2 Configurator

The TC3 IoT Data Agent configurator is an easy-to-use, graphical user interface that abstracts the XML configuration file and provides a modern interface that includes all functionalities to easily configure symbols that should be sent to or received from a cloud service.

The configurator is also used to configure TC3 IoT Functions.



Standard components

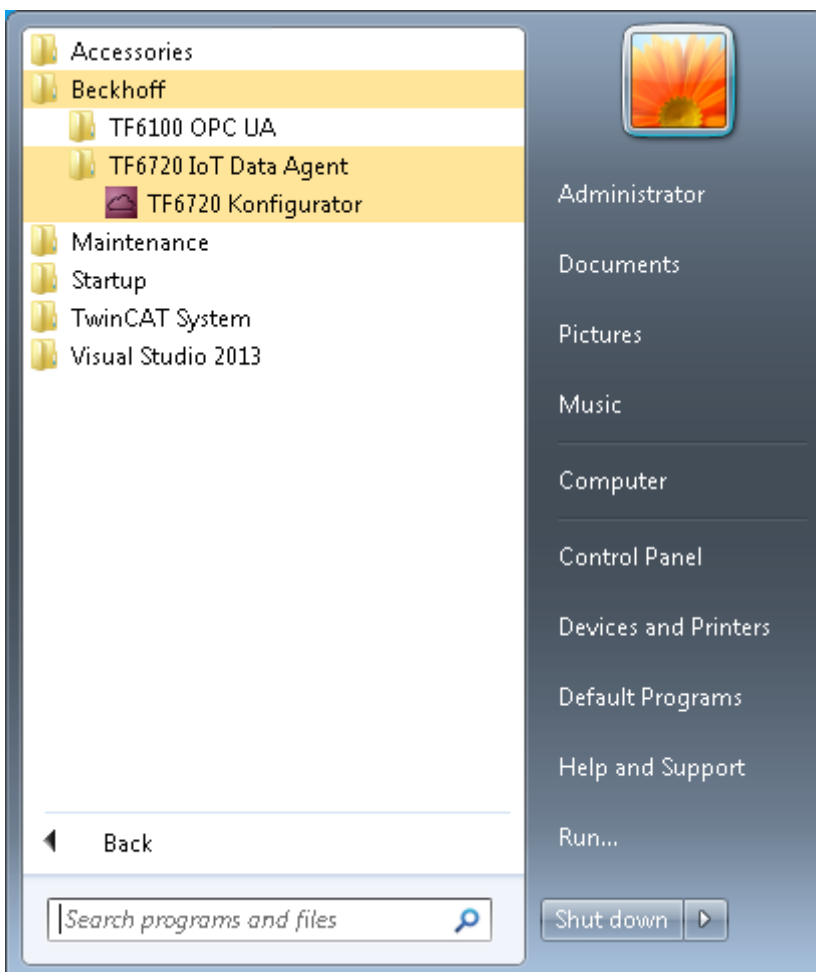
The TC3 IoT Data Agent configurator consists of the following areas:

Menu and toolbar	Provides commands for saving and opening files and for starting and stopping the application
Hierarchical TreeView	Gives a hierarchical overview of the configuration to create and edit a configuration
Topology view	Gives a graphical overview of the configuration to create and edit a configuration
Properties window	Shows the properties of an activated component in the topology or tree view.
Logging	Provides log information from the configurator.
Cascading Editor	Helps to navigate through large and complex navigations by providing filtering mechanisms for symbols
Mapping Links	Gives an overview of all links between symbols in the configuration
Target Browser	Is used to provide symbolic access for ADS and OPC UA target runtimes

Installation

The configurator is automatically installed by the setup and is available as a shortcut on the Windows start menu.

When starting the configurator the first time, it asks for the generation of an OPC UA client certificate. This certificate is used by the configurator in its integrated OPC UA target browser to connect to a server and browse its namespace. After the certificate has been generated, the configurator UI is shown.



5.2.1 Topology view

The topology view (or “canvas”) is the central graphical configuration area of the TC3 IoT Data Agent configurator. It shows the following components of a configuration:

- The configured gates, channels and symbols
- The relationship (mapping) between gates, channels and symbols
- The cardinality on each relationship

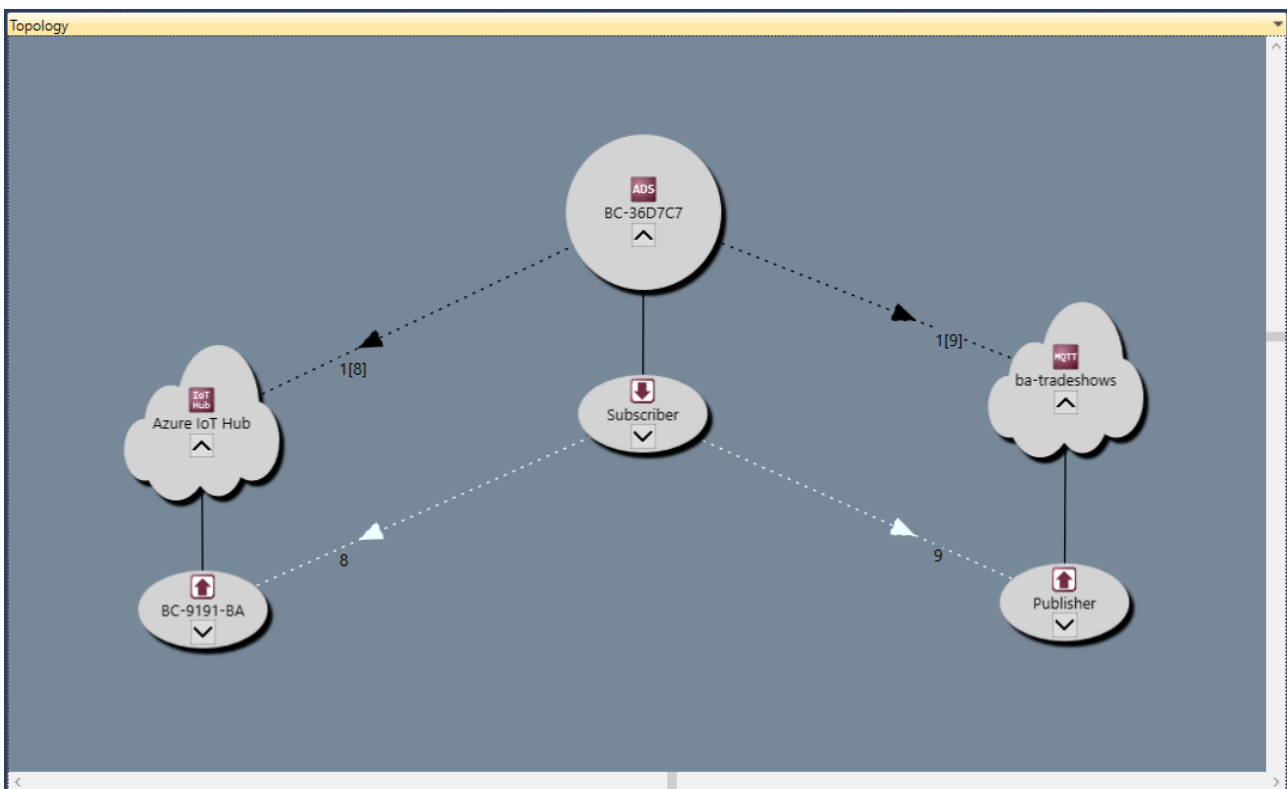
The topology view can be used to create and edit a configuration. Simply right-click the canvas to open the context menu and select from the variety of configuration options, e.g. to create a new gate, attach a new channel to a gate or remove a component from the configuration.

You can move any object in the topology view by dragging it to a new position. Each attached subcomponent is moved together with its parent. Optionally, you can also hide a subcomponent by clicking the expand button of its parent. When saving a configuration, the position of each object is saved in the configuration file.

In order to make navigational tasks a little easier, the topology view supports the following functionalities:

- Scrollbars for vertical and horizontal navigation
- Vertical navigation via mouse wheel
- Horizontal navigation via mouse wheel and SHIFT key (press and hold)
- ZoomIn/ZoomOut via mouse wheel and CTRL key (press and hold)

Instead of the topology view you can also use the tree view for configuration, but the topology view might give a better graphical overview of the currently configuration (see [Tree view \[► 37\]](#))

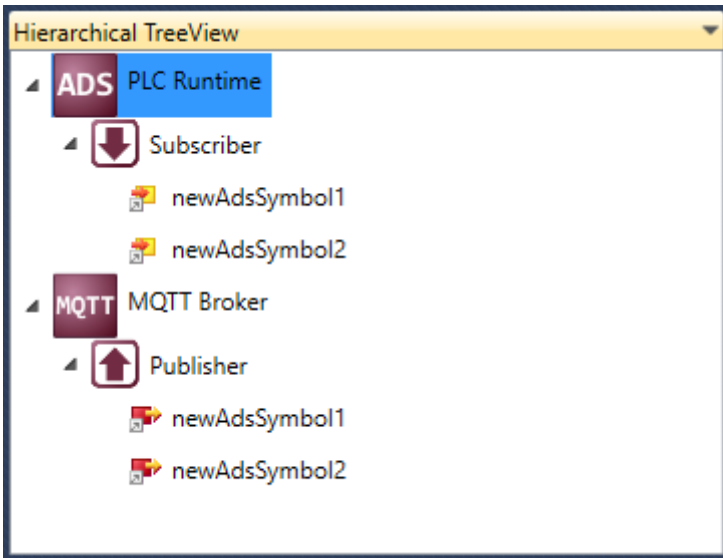


5.2.2 Tree view

The tree view provides a hierarchical view of the currently opened configuration. It shows the following components of a configuration:

- The configured gates, channels and symbols
- The existence of a relationship (mapping) between symbols

The tree view can also be used to edit the configuration, but you might find it easier to use the topology view instead (see [Topology view \[▶ 37\]](#)).



5.2.3 Mappings

The mappings window gives an overview of all links between symbols in the currently opened configuration. When a link is selected, it is automatically highlighted in the topology view.

Icon	Source Gate	Source Channel	Source Symbol	Source DataType	Icon	Destination Gate	Destination Channel	Destination Symbol	Destination DataType	Filter Symbols	Valid	Visible
ADS	BC-36D7C7	Subscriber	TemperatureActual	REAL	PLC	ba-tradeshows	Publisher	TemperatureActual	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureActual	REAL	PLC	ba-tradeshows	Publisher	TemperatureActual	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureSetPointShift_AI00	REAL	PLC	ba-tradeshows	Publisher	TemperatureSetPointShift_AI00	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureSetPoint_AI01	REAL	PLC	ba-tradeshows	Publisher	TemperatureSetPoint_AI01	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	CoolingControlValue_AI02	REAL	PLC	ba-tradeshows	Publisher	CoolingControlValue_AI02	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	HeatingControlValue_AI03	REAL	PLC	ba-tradeshows	Publisher	HeatingControlValue_AI03	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	KeyCardDetected	UINT	PLC	ba-tradeshows	Publisher	KeyCardDetected	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	BalconyDoorOpened	UINT	PLC	ba-tradeshows	Publisher	BalconyDoorOpened	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	PresentDetected	UINT	PLC	ba-tradeshows	Publisher	PresentDetected	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureActual	REAL	PLC	Azure IoT Hub	BC-9191-BA	TemperatureActual	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureSetPointShift_AI00	REAL	PLC	Azure IoT Hub	BC-9191-BA	TemperatureSetPointShift_AI00	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	TemperatureSetPoint_AI01	REAL	PLC	Azure IoT Hub	BC-9191-BA	TemperatureSetPoint_AI01	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	CoolingControlValue_AI02	REAL	PLC	Azure IoT Hub	BC-9191-BA	CoolingControlValue_AI02	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	HeatingControlValue_AI03	REAL	PLC	Azure IoT Hub	BC-9191-BA	HeatingControlValue_AI03	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	KeyCardDetected	UINT	PLC	Azure IoT Hub	BC-9191-BA	KeyCardDetected	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	BalconyDoorOpened	UINT	PLC	Azure IoT Hub	BC-9191-BA	BalconyDoorOpened	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ADS	BC-36D7C7	Subscriber	PresentDetected	UINT	PLC	Azure IoT Hub	BC-9191-BA	PresentDetected	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Filtered View!

Logging Cascading Editor Mapping Links Target Browser Application Config

5.2.4 Target Browser

The Target Browser is used to provide symbolic access for ADS and OPC UA target runtimes. It can be used to configure symbols for a target runtime by drag-and-drop.

The screenshot shows the 'Target Browser' window. On the left, there is a tree view of ports and targets, including '5508973-020', 'CX-124752', 'CX-304858', '350: PlcTask', '851: Port851', 'BC-36D7C7', '4016090-002', and 'SystemService'. The main area displays a table of symbols with the following columns: Name, Type, Size, Category, Comment, Subitems, Unit, Context-Mask, Index-Group, Index-Offset, Attributes (Instance), and Attributes (Type).

Name	Type	Size	Category	Comment	Subitems	Unit	Context-Mask	Index-Group	Index-Offset	Attributes (Instance)	Attributes (Type)
GVL_static_	0	Struct	84				0	0	0	none	
MAIN	0	Struct	9				0	0	0	none	
complex	ST_96	Struct	4				4040	7D53C		none	none
fbTest1	FB_11	Struct	9				0	4040	7D454	<OPC.UA.DA: 1>	none
fbTest2	FB_11	Struct	9				0	4040	7D4C8	none	none
i	INT_2	Primitiv	0				0	4040	7D452	none	none

Logging Cascading Editor Mapping Links Target Browser Application Config

Fig. 1:

Name	Type	Size	Category	Full-Name	Comment	Subitems	NodeClass	Identifier	NamespaceIndex
Views	Struct	0	Views			0	Object	87	0
Objects	Struct	0	Objects			5	Object	85	0
Server	Struct	0	Objects.Server			14	Object	2253	0
Configuration	Struct	0	Objects.Configuration			5	Object	16	7
PLC1	Struct	0	Objects.PLC1			12	Object	PLC1	1
AlarmsConditions	Struct	0	Objects.AlarmsConditions			0	Object	AlarmsConditions	6
DeviceSet	Struct	0	Objects.DeviceSet			1	Object	5001	2
Types	Struct	0	Types			5	Object	86	0

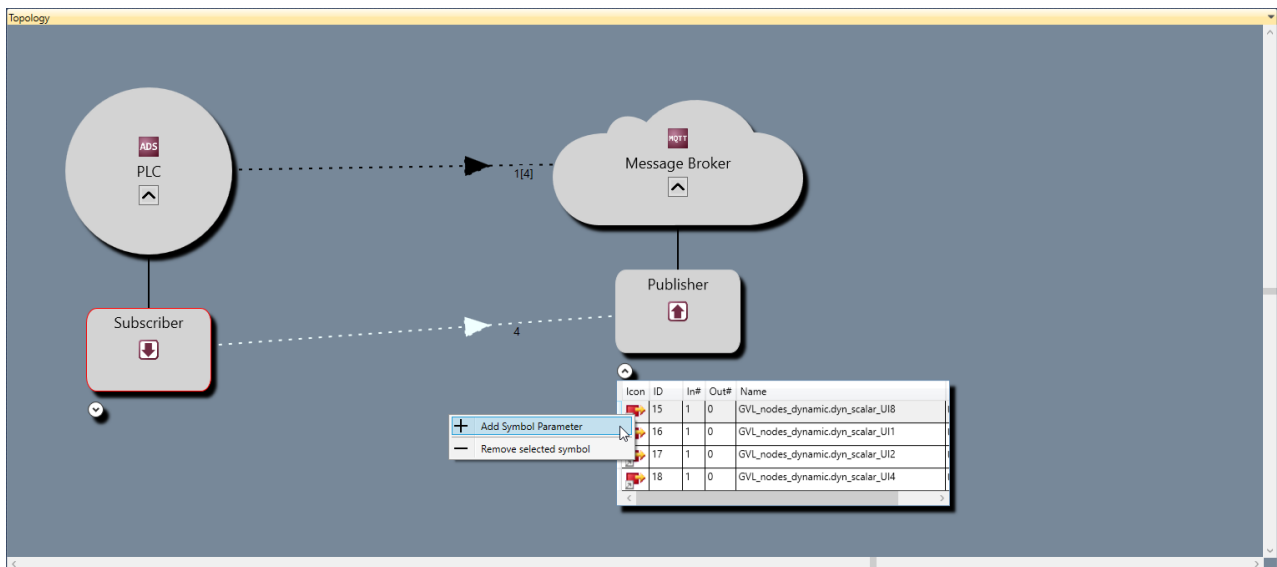
5.2.5 Cascading Editor

The Cascading Editor helps to navigate through large and complex configurations by providing filtering mechanisms for symbols. Starting from left to right you can select gates and channels in order to display the corresponding symbols. In addition, you can also search for symbol names using free text. When a component is selected, it is automatically highlighted in the topology view.

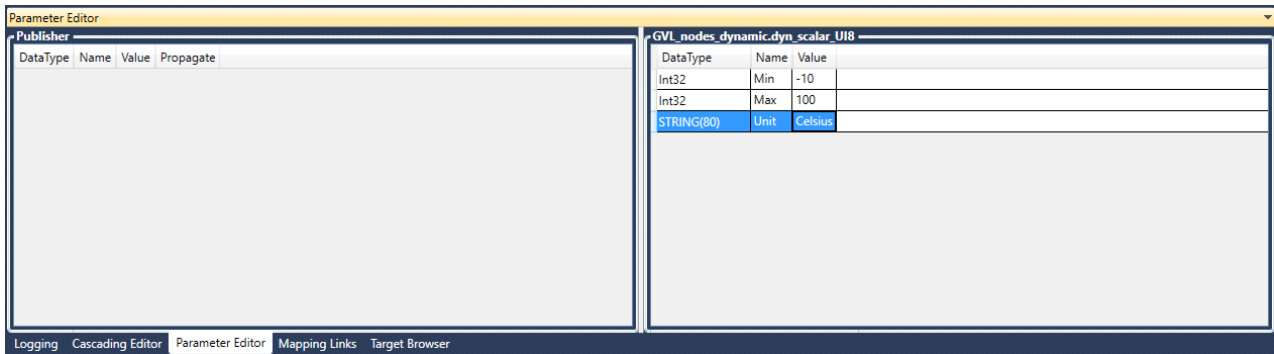
Icon	Id	URN	Data Type	Conversion	Channel	Gate	#In	#Out	Para
	5	newAdsSymbol1	NONE	Lossless	Subscriber	PLC Runtime	0	1	0
	6	newAdsSymbol2	NONE	Lossless	Subscriber	PLC Runtime	0	1	0

5.2.6 Parameter Editor

The Parameter Editor allows to configure symbol meta data for symbols in a publisher channel. As a prerequisite, the publisher channel has to be configured to use the "TwinCAT JSON" data format. You can then add new symbol parameters by right clicking a symbol and select **Add Symbol Parameter**.



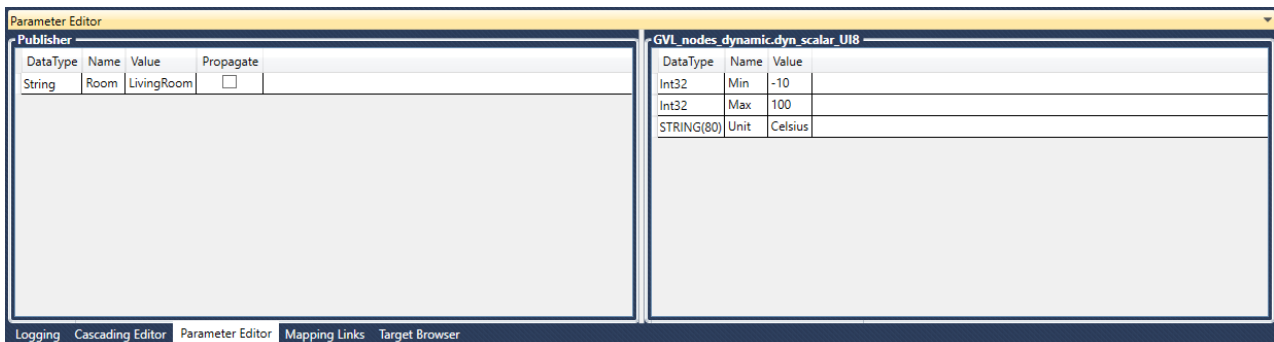
This will add a new symbol parameter entry to the Parameter Editor. You can then specify the data type, name and value of the new parameter.



A result of the above configuration may look as follows when using the TwinCAT JSON data format. In this configuration, the variable "GVL_nodes_dynamic.dyn_scalar_UI8" has been configured with the symbol parameters Min, Max and Unit. These parameters will be added to the "MetaData" section of the TwinCAT JSON formatted message.

```
{
  "Timestamp": "2020-09-28T12:32:49.0370000+02:00",
  "GroupName": "Publisher",
  "Values": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": 259096147,
    "GVL_nodes_dynamic.dyn_scalar_UI1": 83,
    "GVL_nodes_dynamic.dyn_scalar_UI2": 32339,
    "GVL_nodes_dynamic.dyn_scalar_UI4": 259096147
  },
  "MetaData": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00",
      "Min": -10,
      "Max": 100,
      "Unit": "Celsius"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI1": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI2": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI4": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    }
  }
}
```

In addition, you can also add parameters to a channel, which will add meta data properties to the root message. Example:



As a result, the root message now contains a static meta data property called "Room", which has the value "LivingRoom":

```
{
  "Timestamp": "2020-09-30T09:55:58.1160000+02:00",
  "GroupName": "Publisher",
  "Room": "LivingRoom",
  "Values": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": 267266470,
    "GVL_nodes_dynamic.dyn_scalar_UI1": 166,

```



```

"GVL_nodes_dynamic.dyn_scalar_UI2": 10662,
"GVL_nodes_dynamic.dyn_scalar_UI4": 267266470
},
"MetaData": {
  "GVL_nodes_dynamic.dyn_scalar_UI8": {
    "Timestamp": "2020-09-30T09:55:58.1140000+02:00",
    "Min": -10,
    "Max": 100,
    "Unit": "Celsius"
  },
  "GVL_nodes_dynamic.dyn_scalar_UI1": {
    "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
  },
  "GVL_nodes_dynamic.dyn_scalar_UI2": {
    "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
  },
  "GVL_nodes_dynamic.dyn_scalar_UI4": {
    "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
  }
}
}
}

```

5.2.7 Settings

This section provides some detailed information about the different configuration parameters that can be set on gates, channels and symbols.

5.2.7.1 Gates

A Gate represents a communication protocol or specific connectivity service, e.g. ADS, OPC UA, MQTT, AWS IoT or Microsoft Azure IoT Hub. Each Gate is configured with parameters that are specific for the corresponding Gate type.

ADS

A Beckhoff ADS device represents either a TwinCAT 2/3 or a Beckhoff BC device. ADS is the common Beckhoff communication protocol and can be used to access many parts of the TwinCAT system. The most common application scenarios for TC3 IoT Data Agent involve access to a TwinCAT 2/3 PLC, C++, TcCOM modules or the I/O process image, which is done via ADS.

When configuring an ADS gate, the following settings are required by the TC3 IoT Data Agent to access the underlying ADS device:

Setting	Description
AmsNetId	AmsNetId of the target device, e.g. 127.0.0.1.1.1 for the local device.
AdsPort	AdsPort of the target device, e.g. 801 (TwinCAT 2 PLC) or 851 (TwinCAT 3 PLC)
IoMode	Specifies how the TC3 IoT Data Agent should communicate with the ADS device. The following options can be set: <ul style="list-style-type: none"> • Direct: accesses every symbol with a separate ADS command. Mandatory for BC controllers but increases ADS traffic • Batched: accesses symbols batched into an ADS sum command, which optimizes ADS traffic and can be used for TwinCAT 2 and 3 PLC or C++ runtimes

OPC UA

OPC UA is a standardized, industrial, client/server communication protocol and adopted by many vendors for different use cases. The TC3 IoT Data Agent can access OPC UA server devices to connect variables (so-called “nodes”) on those devices with IoT services.

When configuring an OPC UA gate, the following settings are required by the TC3 IoT Data Agent to access the underlying OPC UA device:

Setting	Description
Server URL	The OPC UA Server URL, e.g. opc.tcp://localhost:4840
Security policy	The OPC UA security policy that the TC3 IoT Data Agent should use during connection establishment with the OPC UA server
Security mode	The OPC UA message security mode that the TC3 IoT Data Agent should use during connection establishment with the OPC UA server
Authentication mode	The OPC UA authentication mode that the TC3 IoT Data Agent should use during connection establishment with the OPC UA server

MQTT

MQTT can be used for connecting to a generic message broker, e.g. Mosquitto, HiveMQ or similar broker types.

Setting	Description
Broker address	The IP address or hostname of the MQTT message broker
Port	MQTT specifies port 1883 for unencrypted communication and 8883 for encrypted communication
ClientId	A numeric or string-based value that identifies the client. Depending on the message broker type, this ID should be unique
Authentication mode	Specifies if the TC3 IoT Data Agent should authenticate to the broker by using a username/password combination
TLS mode	Specifies if TLS should be used to secure the communication channel to the message broker. Note that the message broker also needs to support TLS in order for this to work. Different options are available for TLS: <ul style="list-style-type: none"> • CA certificate: Only uses the CA certificate for server authentication • Client certificate: Uses a client certificate for mutual client/server authentication • PSK: Used a common PSK-Identity and PSK-Key that is known to the message broker and the client

Microsoft Azure IoT Hub

With Azure IoT Hub the Microsoft Azure cloud platform offers a connectivity service in the cloud that provides bi-directional communication, device security and automatic scalability. In the TC3 IoT Data Agent, the IoT Hub can be configured as a special gate type.

Setting	Description
HostName	URL of the Azure IoT Hub instance
DeviceId	DeviceId of the device that has been created on the IoT Hub configuration website
SharedAccessKey	Either the primary or secondary device key that has been generated together with the device on the IoT Hub configuration website
CA file	CA file that is used for server authentication. At the time writing this article, three certificates are in play during the server authentication, which form part of the certificate chain and are linked together. <ul style="list-style-type: none"> • Root CA: Baltimore CyberTrust Root • Intermediate CA: Baltimore CyberTrust • Wilcard certificate <p>During the initial TLS handshake, only the first two are sent by the server to the client. The client will normally only validate the Root CA of the chain and will determine if it is trusted.</p> <p>In order to acquire the CA file for the Root CA, you can open the Windows certificate store (certmgr.msc), browse to the trusted root authorities and export the Baltimore CyberTrust Root certificate.</p>

AWS IoT

With AWS IoT the Amazon Web Services cloud platform offers a message broker service in the cloud that provides bi-directional communication, device security and automatic scalability. In the TC3 IoT Data Agent, AWS IoT is configured as a regular MQTT gate. However, some special MQTT settings have to be configured to successfully connect to an AWS IoT instance.

Setting	Description
Broker address	The URL of the AWS IoT instance
Port	Port 8883 for encrypted TLS communication is mandatory
ClientId	Can be set to anything but needs to be unique. Typically this could be the AWS IoT thing name.
Authentication mode	Set to "No authentication". Authentication on AWS IoT is done via the TLS client certificate.
TLS mode	Automatically set to "Client certificate" by the configurator. Select path to CA file, client certificate file and client key file. These are the files that can be generated and downloaded on the AWS IoT configuration website.

5.2.7.2 Channels

Channels are configured on a Gate to send ("publisher") or receive ("subscriber") data to/from a Gate.

- Source: This Gate is the source of data, meaning the TC3 IoT Data Agent connects to the Gate and retrieves data from it in order to send this data somewhere else (to a "Destination Gate"). Technically, this is also referred to as the "Subscriber Channel".
- Destination: This Gate is the target of data, meaning the TC3 IoT Data Agent connects to the Gate and sends data to it, which has been acquired from another Gate (from a "Source Gate"). Technically, this is also referred to as the "Publisher Channel".

Every channel has different settings that either describe the data format that should be used for this channel or the sampling settings that the TC3 IoT Data Agent should use for gathering data. These settings can also depend on the gate type that the channel has been configured for.

The following tables list all available settings.

Setting	Description	Applicable to Gate type
Direction	Sets whether the channel should either be a publisher (sender) or subscriber (receiver) channel. Depending on the selection and gate type, other settings are required or pre-selected.	All Gates

Setting	Description	Applicable to Gate type
SamplingMode	Selects if the channel should use either cyclic or event-based sampling mechanisms when gathering data from a source. Note that, depending on the direction, not all Gates might support both types. An MQTT Gate, for example, uses when receiving data always SamplingMode "event" (because of the Pub/Sub principle it is always event-based).	All Gates
CycleTime	Only applicable for SamplingMode "cyclic". Sets the sampling rate in [ms].	All Gates
Timeout	The timeout for a communication with the Gate in [ms].	All Gates
PartialUpdate	Enable/Disable partial updates on this channel. When enabled (default), a publish includes only the updated symbol. When disabled, a publish includes all symbols of a channel with their last known value. Only applicable to publisher channels.	All Gates
BufferSize	Sets the size (amount of messages) of the ringbuffer on connection loss.	MQTT, AWS IoT, Azure IoT Hub

Setting	Description	Applicable to Gate type
Formatter	Sets the data format that should be used for this channel, e.g. binary or JSON. Note that some Gates require their Channels to use a fixed data format, e.g. ADS or OPC UA Gates, because the communication with those devices requires a specific format. In this case the Formatter is fixed and not changeable via the configurator.	MQTT, AWS IoT, Azure IoT Hub
FormatterType	Sets the formatter type on this channel. In most cases the formatter type is an InOut type. Consult our documentation article about writing custom plugins via the formatter interface for more information about this setting.	MQTT, AWS IoT, Azure IoT Hub

Setting	Description	Applicable to Gate type
Topic	Specifies the MQTT topic that should be used to publish or subscribe to.	MQTT
QoS	Specifies the <u>QoS (Quality-of-Service)</u> level that should be used when publishing or subscribing.	MQTT, AWS IoT
Retain	Specifies if a message should be send as retain. (only relevant for publisher channel)	MQTT
SendStateInfo	<p>When activated, the TC3 IoT Data Agent publishes its "OnlineState" to the sub topic /Desc/ as well as uses this sub topic in its LastWill. When the TC3 IoT Data Agent connects to the Message Broker, a JSON message will be published to this topic, which contains</p> <pre>{ "OnlineState" : true }</pre> <p>When the TC3 IoT Data Agent disconnects orderly from the Message Broker, the following message is sent to this topic:</p> <pre>{ "OnlineState" : false }</pre> <p>When the Message Broker detects, that the TC3 IoT Data Agent lost connection, the following message is sent to this topic (LastWill):</p> <pre>{ "OnlineState" : false }</pre>	MQTT, AWS IoT

Sampling Modes

The TC3 IoT Data Agent includes different sampling modes that influence the way data is acquired from a source or written to a destination. The sampling mode can be set on a channel. The following sampling modes are currently available:

- Cyclic
- OnChange
- TriggerSymbol

Cyclic

Cyclic sampling means that the TC3 IoT Data Agent is cyclically sampling the gate for data (subscriber channel) or cyclically writing data to a gate (publisher channel). On a subscriber channel, this results in cyclic read commands whereas on a publisher channel this results in cyclic write commands, e.g. on an ADS or OPC UA gate. On gate types that are based on publisher/subscriber concepts, e.g. MQTT, AWS IoT and Azure IoT Hub gates, cyclic requests on a subscriber channel are automatically replaced by subscriptions whereas on a publisher channel this results in cyclic publish commands.

OnChange

OnChange sampling means that the TC3 IoT Data Agent only communicates data with a gate if the value of a variable has changed.

Trigger symbols

Trigger symbols enable “on demand” sampling, e.g. if a certain condition for a specified symbol (the so-called “trigger symbol”) is fulfilled. Different condition types can be set. They are configured as part of a channel and allow to specify the following condition types.

Condition type	Description
EQ	Trigger symbol value equals a given value
NE	Trigger symbol value differs from a given value
LE	Trigger symbol value is less than or equals a given value
GE	Trigger symbol value is greater than or equals a given value
LT	Trigger symbol value is less than given value
GT	Trigger symbol value is greater than given value

If the condition is fulfilled, all symbols in that channel are published to the corresponding gate. In addition you can specify how often the symbol values should be send.

Send behavior	Description
risingEdge	Symbols are only send once when the condition is fulfilled
continuous	Symbols are send as long as the condition is fulfilled

● Usage of trigger symbols



Trigger symbols can only be configured for ADS and OPC UA subscriber symbols.

- Add the symbol that should act as trigger symbol to the ADS or OPC UA subscriber channel.
- Configure the linked publisher channel with SamplingMode “OnTrigger” to specify the previously added symbol as trigger symbol.

5.2.7.3 Symbols

Symbols represent variables from a gate, e.g. a TwinCAT PLC variable. The symbol configuration includes the address information that the Data Agent requires in order to read a symbol’s value or write to it. This address information is therefore depending on the gate type.

On gates that support a [target browser \[▶ 38\]](#), the browser will automatically detect the correct address information for a symbol. For all other gates, this information needs to be entered manually.

Gate type	Setting	Description
ADS	URN	Symbol name address information of the ADS variable. Might not work with all ADS devices, e.g. BC devices do not support symbol names.
ADS	IndexGroup IndexOffset	IndexGroup/IndexOffset combination can be used to access data of an ADS device that does not support symbolic address information. In case of the TwinCAT PLC, the IndexGroup/IndexOffset combination directly represents a memory address, e.g. from a PLC variable, which may change after a re-compilation of the TwinCAT project. It is therefore common practice to use the TwinCAT PLC symbol server instead, which provides symbolic information for its PLC variables, which means that a variable can be accessed via its symbol name instead, which stays valid even after a re-compilation or online change (if the symbol still exists). However, some ADS services do not include such a symbol server, e.g. small Beckhoff BC devices. In those cases the IndexGroup/IndexOffset combination needs to be used.
ADS	DataType	Data type of the symbol
ADS	Conversion	Specifies conversion mode for this symbol.
OPC UA	Name	Descriptive name of the OPC UA node. Only used in Configurator, does not represent any online address information.
OPC UA	Identifier	Identifier of OPC UA symbol on the server, e.g.: <ul style="list-style-type: none"> • s = MAIN.nCounter (if the IdentifierType is "String") • n = 42 (if the IdentifierType is "Numeric")
OPC UA	NsName	Namespace name in which the symbol is located. This corresponds to the namespace index, which is part of an OPC UA NodeId. The translation can be done via the NamespaceArray.
OPC UA	AttributeId	The AttributeId defines the OPC UA attribute that should be used by the Data Agent when reading a symbol. In most scenarios, this is the "Value" of a symbol.
OPC UA	Conversion	Specifies conversion mode for this symbol.
MQTT	URN	Name of the MQTT symbol. This represents the name that is being used in the JSON format as a key, also when receiving data from the MQTT Gate.
MQTT	DataType	Data type of the symbol
MQTT	Conversion	Specifies conversion mode for this symbol.
IoT Hub	URN	Name of the IoT Hub symbol. This represents the name that is being used in the JSON format as a key, also when receiving data from IoT Hub.
IoT Hub	DataType	Data type of the symbol
IoT Hub	Conversion	Specifies conversion mode for this symbol.

When configuring a Channel, symbols can be added via a target browser or manually by providing the correct address information. Note that not all gate types include target browser functionalities. In this case symbols need to be configured manually.

Manual symbol configuration

If the target device is not online or does not provide any symbolic address information (e.g. the BC9191), symbols may also be added manually by entering the symbol address. The tables at the beginning of this document show which information is required in this case.

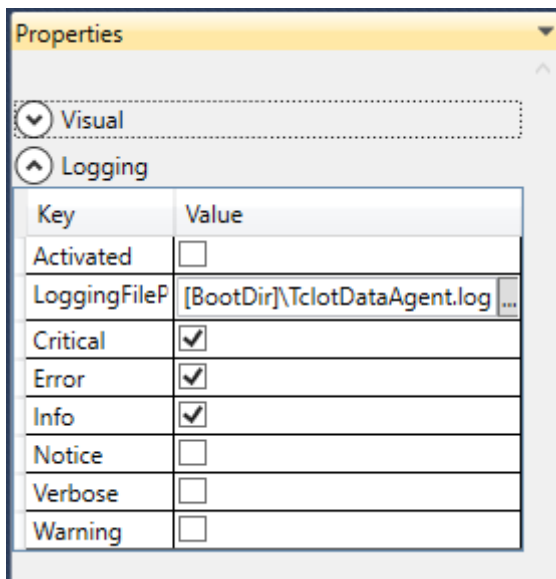
Type conversion

The TC3 IoT Data Agent supports data type conversion before the data is published to a Gate. Type conversion occurs on symbol level, which means that different Symbols can use different conversion modes. The following table shows the different conversion modes available.

Conversion mode	Description
Losless	Default setting. "Smaller" types can be converted into "larger" types. For example, a subscriber symbol of data type INT can be published to a symbol of data type Int32 (2 byte to 4 byte).
Lossy	If required, "larger" symbols can also be converted into "smaller" symbols. For example, a subscriber symbol of data type DINT can be published to a symbol of data type Int16 (4 byte to 2 byte). Depending on the value of the subscriber symbol, this can of course result in cut-off values.
Strict	If required, symbols can also be configured to use "strict mode". In this conversion mode, the data type size of a subscriber symbol needs to match exactly the data type of the mapped publisher symbol. For example, a subscriber symbol of data type INT can only be published to a symbol of data type Int16 (2 byte to 2 byte).

5.2.8 Error logging

For troubleshooting purposes, the TC3 IoT Data Agent can generate a logfile, which can be populated based on different logging levels. All necessary settings can be configured via the properties window of the configurator. Simply click on an empty spot on the topology view and configure the logging settings in the properties window.



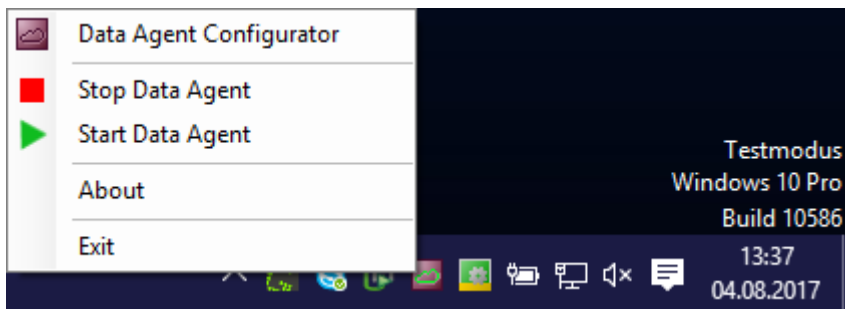
Note that all settings are tied to the currently opened configuration and have to be set individually for every configuration. The default directory in which the logfile is created, is the TwinCAT boot directory. The placeholder [BootDir] automatically selects the TwinCAT boot directory.

● Logging window




i The logging window inside of the configurator is only showing log events that are related to the configurator. In order to create a log for the TC3 IoT Data Agent background service, the settings above are required.

5.3 System tray

Local TC3 IoT Data Agent functionalities are also available via the Windows system tray, e.g. starting or stopping the locally running TC3 IoT Data Agent. The TF6720 setup automatically installs a system tray application that provides these functionalities via the context menu.



In addition, the system tray icon shows the current state of the TC3 IoT Data Agent.

Mode	Symbol	Description
Idle mode		No configuration is currently loaded and active
Run mode		A configuration is currently loaded and active
Stop mode		A configuration is loaded but currently not active, e.g. due to an error

5.4 Support Information Report

The Support Information Report is a tool for collecting product information for submission to Beckhoff technical support. Collecting product-related data such as TwinCAT version/build, product version, image version and device type reduces email traffic significantly and enables more efficient advice.

Plug-in mechanism

Various Beckhoff products interface with the Support Information Report via a plug-in mechanism. These products, such as the TwinCAT Database Server, have a Support Information Report entry in the corresponding product menu.

Creating and submitting a Support Information Report

- ✓ A Support Information Report is open.
- 1. Use the **Behaviour** text field to describe the behavior that occurred in as much detail as possible.
- 2. In the **Attachment** area, you can add files (screenshots etc.) to the report via the **Add Attachment** button, if required. Files can optionally be selected via remote access. To do this, select a target from the **Remote System** dropdown list. Depending on the selected target, it may be possible to browse Windows CE devices.
- 3. Enter your contact details and select a Beckhoff subsidiary for your country. This information is obligatory for submitting the Support Information Report.
- 4. You will be offered the option to store your contact details for future Support Information Reports. To do this, tick the **Store personal data** check box.
- 5. The product-specific plug-ins can be found in the lower section of the Support Information Report. Tick the **Include in report** check box. The information required for the product is added automatically, if it is available. The screenshot shows the current configuration of a TwinCAT Database Server in the form of an XML file as an example.

6. Submitting the Support Information Report:

- If the device has an email connection, you can submit the Support Information Report directly to the Beckhoff subsidiary for your country via the **Send Report** button.
- If the device does not have an email connection, you can save the Support Information Report locally as a .zip file via the **Save .zip** button and then make it available via FTP, USB etc.

Send Report Save .zip

Information

Behaviour

1

Remote System: Local - 172.17.214.70.1.1

Attachment Add Attachment

2

Personal Data 3

Name: Max

Lastname: Mustermann

Company: Beckhoff Automation GmbH

Your Country: Germany

City: Verl

Street: Hülshorstweg 20

Phone: +49 5246 9630

e-Mail: support@beckhoff.de

Beckhoff subsidiary country: Germany

Store personal data

TC DbSrv TC Scope

Include in report: 4

Attachments:

C:\TwinCAT\3.1\Boot\CurrentConfigDataBase.xml

6 Samples

This section provides configuration examples that demonstrate how to use the TwinCAT IoT Data Agent to connect to specific device types and cloud services. The section is categorized into "Southbound" and "Northbound" samples.

- Southbound: connections to underlying devices, e.g. via ADS to TwinCAT or OPC UA to third-party devices.
- Northbound: connections to cloud services

Also see about this

📖 Application examples [▶ 17]

6.1 Quick Start

This section describes how to get started quickly with the TC3 IoT Data Agent.

The communication scenario is a typical use case for the Data Agent: Variables from a southbound device (TwinCAT PLC runtime) are sampled and published to a northbound service (MQTT Message Broker). In this example, the message broker is running on the same computer as the PLC runtime.

To set up this scenario, the following steps are required. All steps are performed on the same computer.

- [Install the software \[▶ 50\]](#)
- [Prepare the PLC program \[▶ 50\]](#)
- [Configure the TC3 IoT Data Agent \[▶ 51\]](#)
- [Display the published messages \[▶ 52\]](#)

Install the software

1. Install TF6720 IoT Data Agent as described in the [Installation \[▶ 10\]](#) section.
2. Install Mosquitto MQTT Message Broker with its default setup settings. The Message Broker can be installed anywhere and is connected to via its hostname or IP address.
3. Open a Windows Console, navigate to the installation directory of the Mosquitto MQTT Message Broker (typically *C:\Program Files (x86)\mosquitto*) and start the Broker in verbose mode by executing the following command: `mosquitto.exe -v`

● Mosquitto MQTT Message Broker



The Mosquitto MQTT Message Broker can be downloaded from www.mosquitto.org. Make sure to follow any installation instructions provided by the Mosquitto installer.

● Mosquitto startup



By default, the Mosquitto MQTT Message Broker is installed as a Windows service. You can also start the `mosquitto.exe` from a command line and specify parameters like `"-v"` for verbose output. If you want to start the broker from a command line, make sure that the Mosquitto Windows service is not running.

Prepare the PLC program

Set up a TwinCAT 3 PLC project and activate trial licenses for TC1200 (PLC) and TF6720 (TC3 IoT Data Agent). In this sample the PLC program should include the following variables:

```
PROGRAM MAIN
VAR
  nCounter1 : INT;
  nCounter2 : LREAL;
END_VAR
```

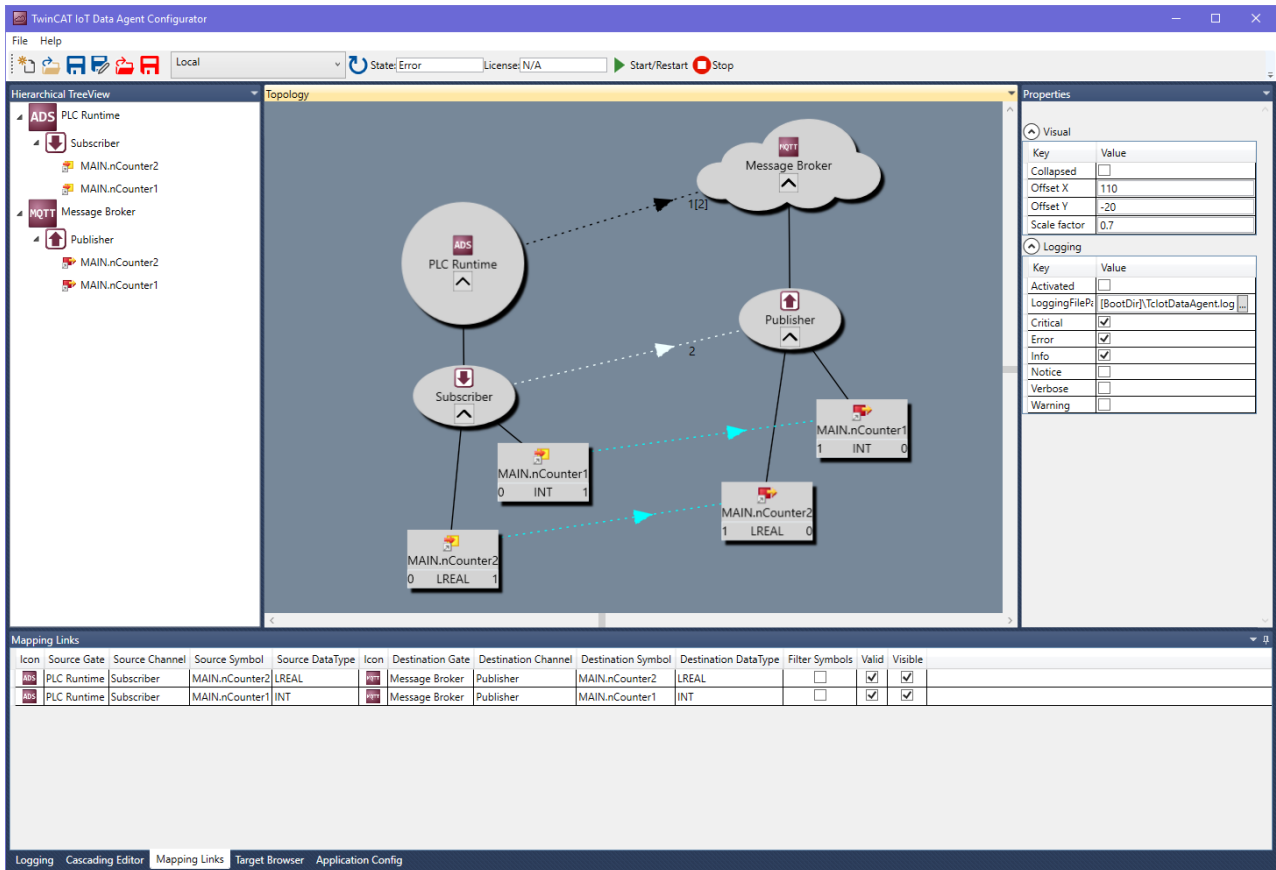
These two variables should be incremented cyclically. Add the following implementation code for the PLC program:

```
nCounter1 := nCounter1 + 1;  
nCounter2 := nCounter2 + 0.1;
```

Activate the project and start TwinCAT into run mode. Make sure that the PLC program is running by logging in.

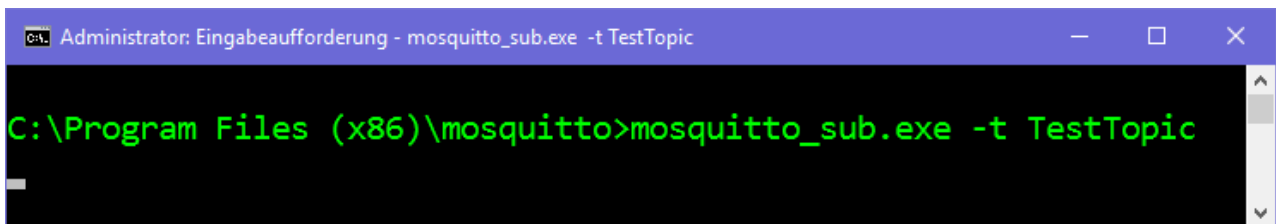
Configure TwinCAT IoT Data Agent

1. Start the TwinCAT IoT Data Agent configurator, either via its link on the Windows start menu or via the system tray application (see [Configurator \[► 35\]](#)).
2. Follow the step-by-step instructions below to create a Data Agent configuration. As an alternative you can also open the configuration file "Sample_QuickStart.xml", which can be found in the Data Agent installation directory, and directly move on with step (m).
3. Start a new configuration by clicking on the **New** icon in the configurator toolbar.
4. In the topology view, right-click a blank area and select **Add Gate (ADS)**.
5. Select the new gate and make sure that the local AmsNetId and ADS port 851 are selected in the properties window
6. In the topology view, right-click the new gate and select **Add Channel (Subscriber)**.
7. Select the new channel and make sure that sampling mode "cyclic" and cycle time "1000" are selected in the properties window.
8. Open the target browser and navigate to your local PLC runtime (port 851).
9. Select the variables "nCounter1" and "nCounter2" and drag them to the subscriber channel in the topology view.
10. In the topology view, right-click a blank area and select **Add Gate (MQTT)**.
11. Select the new gate and make sure that the URL is set to 127.0.0.1 and Port 1883 is being used.
12. In the topology view, right-click the new gate and select **Add Channel (Publisher)**.
13. Select the new channel and make sure that the sampling mode is set to "cyclic". Set the cycle time to "1000" and the formatter to "TwinCAT JSON". The MQTT topic can be set to a topic of your choice, e.g. "TestTopic".
14. Hold the CTRL key and drag the subscriber channel to the publisher channel in order to create a mapping between the source and destination symbols. Note that the destination symbols are automatically created.
15. Activate the configuration by clicking on the **Activate Configuration** button in the toolbar.

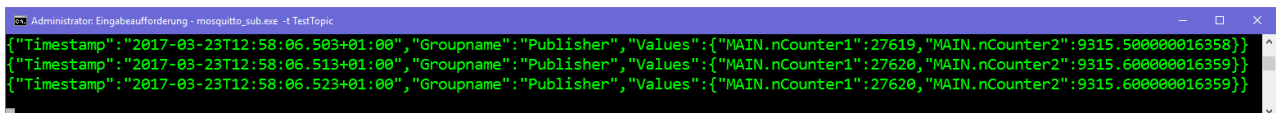


Display the published messages

Open a command line window and navigate to the installation directory of the Mosquitto MQTT Message Broker (typically "C:\Program Files\mosquitto" or "C:\Program Files (x86)\mosquitto"). Start the Mosquitto Subscriber tool by executing the following command:
 mosquitto_sub.exe -t TestTopic.



The Mosquitto Subscriber tool will now receive the messages that are published from the TwinCAT IoT Data Agent.



6.2 Southbound

6.2.1 Connect to third-party devices via OPC UA

Most of the step-by-step instructions in this sample section are based on an OPC UA subscriber channel, but the same principles can also be applied to data acquisition from another device type.

The following section describes step-by-step how to configure the TC3 IoT Data Agent to sample data from an OPC UA Server and to publish it to a MQTT Message Broker by using a JSON data format.

Configure OPC UA gate, channel and symbols

Follow the step-by-step instructions to create an OPC UA gate and a subscriber channel and add OPC UA nodes to the channel's configuration.

✓ The topology view is activated.

1. Right-click the canvas and select **Add Gate (OPC UA)**.
2. Select the new gate and configure OPC UA specific settings in the properties window (Server URL, Security Policy, Message Security Mode, ...).
3. Right-click the new gate and select **Add Channel (Subscriber)**.
4. Select the new channel and set "SamplingMode" to "OnChange". Leave all other fields on their default settings.
5. Open the Target Browser.
6. Navigate to the OPC UA namespace and add some symbols to the configuration by dragging them to the subscriber channel.

⇒ A OPC UA gate and a subscriber channel have been created. OPC UA nodes have been added to the configuration.



ServerURL

Make sure that the ServerURL that you specified in the gate settings equals the ServerURL that is used in the target browser.

Configure MQTT gate and channel

Follow the step-by-step instructions to create a MQTT gate and a publisher channel.

✓ The topology view is activated.

1. Right-click the canvas and select **Add Gate (MQTT)**.
2. Select the new gate and configure MQTT-related settings in the properties window (Broker address, ...).
3. Right-click the new gate and select **Add Channel (Publisher)**.
4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Topic" to "TestTopic". Leave all other fields on their default settings.

⇒ A MQTT gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

The resulting configuration looks like this:

The screenshot shows the TwinCAT IoT Data Agent Configurator interface. The main area displays a topology diagram with the following components and connections:

- OPC UA Server** (left) connected to **MQTT Broker** (right) via a dashed arrow labeled "1[2]".
- Subscriber** (left) connected to **Publisher** (right) via a dashed arrow labeled "2".
- newOpcUASymbol1** (left) connected to **newOpcUASymbol1** (right) via a dashed arrow labeled "1".
- newOpcUASymbol2** (left) connected to **newOpcUASymbol2** (right) via a dashed arrow labeled "1".

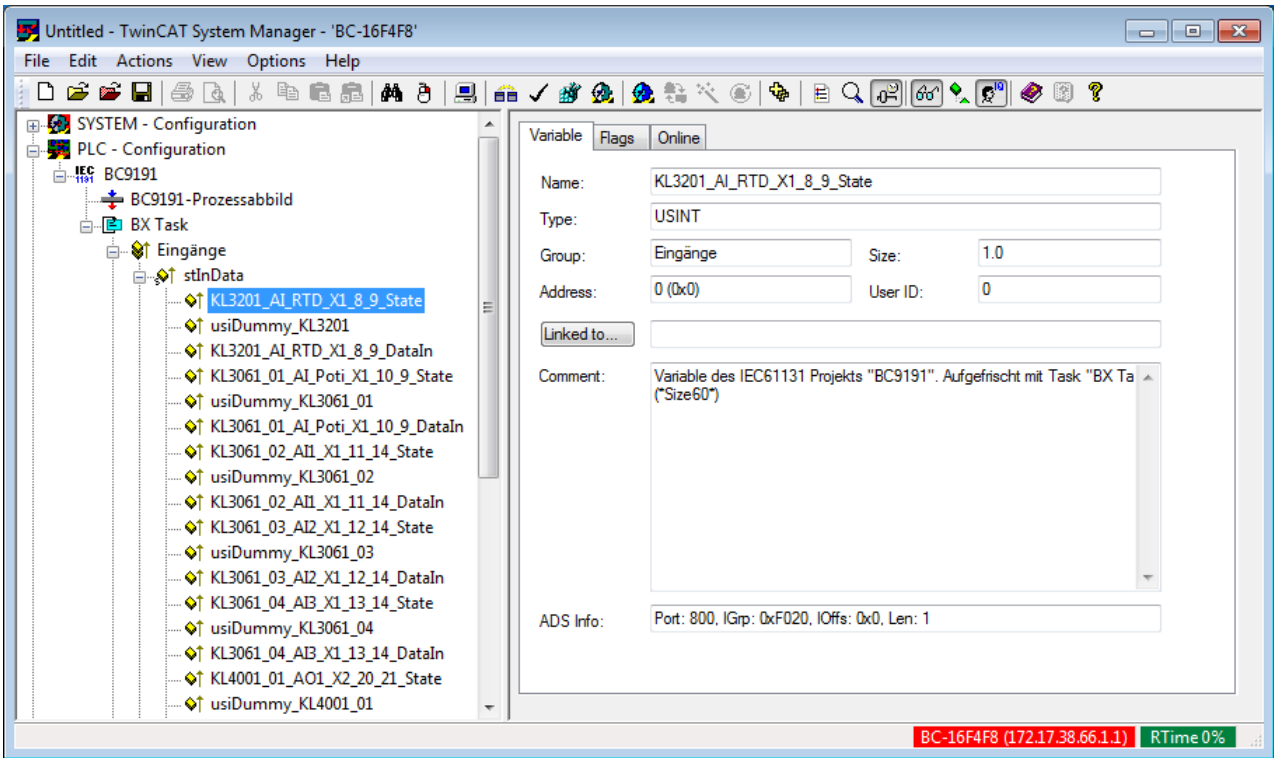
The logging table at the bottom contains the following data:

LogDictionary	RelatedShapeltems	Timestamp	Summary	Description	Priority	OriginCallerMemberName
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterSymbol	newMqttSymbol_4 [DataAgentTopology\View.VMs.SymbolMqttVM]	INFO	RegisterSymbolVM
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterMappingLink	newOpcUASymbol2 to newOpcUASymbol2	INFO	RegisterMappingLinkVM
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterGateLinkVM		INFO	RegisterGateLinkVM
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterChannelLinkVM		INFO	RegisterChannelLinkVM
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterMappingLink	newOpcUASymbol1 to newOpcUASymbol1	INFO	RegisterMappingLinkVM
(Auflistung)	(Auflistung)	2/19/2018 9:07:30 AM	RegisterSymbol	newMqttSymbol_3 [DataAgentTopology\View.VMs.SymbolMqttVM]	INFO	RegisterSymbolVM

6.2.2 Connect to a BC9191 via ADS

In order to configure the TC3 IoT Data Agent to access a BC9191 controller via ADS, the symbols must be configured manually in the TC3 IoT Data Agent configurator, since the BC9191 controller does not provide any symbolic information via ADS. The target browser mechanism cannot be used. The IndexGroup / IndexOffset information must be configured for each symbol to be configured for IoT communication.

In case of input/output variables, this information can be found in the “ADS Info” part in the TwinCAT System Manager.



In case of internal variables, these variables can be configured for ADS access in different ways. Consult the BC9191 documentation for more information.

The following PLC code snippet provides an example:

```
VAR_GLOBAL
  iCounter1 AT%MB100 : INT;
  iCounter2 AT%MB200 : INT;
  iCounter3 AT%IB100 : INT;
  iCounter4 AT%IB200 : INT;
END_VAR
```

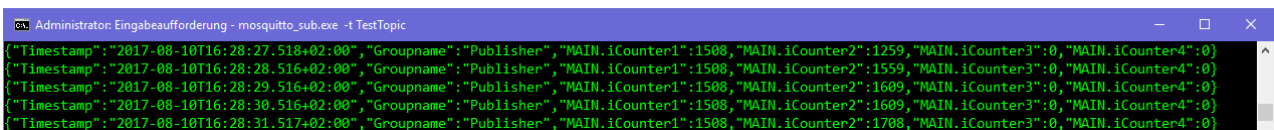
The corresponding IndexGroup/IndexOffset combinations for these two variables are:

Variable	IndexGroup	IndexOffset
iCounter1	0x4020 (16416)	0x64 (100)
iCounter2	0x4020 (16416)	0xC8 (200)
iCounter3	0xF020 (61472)	0x64 (100)
iCounter4	0xF020 (61472)	0xC8 (200)

The BC9191 can be added as a regular ADS gate to the TC3 IoT Data Agent configuration. The ADS port must be set to 800, the IoMode is then automatically set to "direct".

As described, the symbols must be added manually using the corresponding IndexGroup/IndexOffset information. The URN can be set to a spoken name of your choice.

After mapping these subscriber symbols to corresponding publisher symbols, e.g. on an MQTT gate, the variables are shown on the publisher's side.

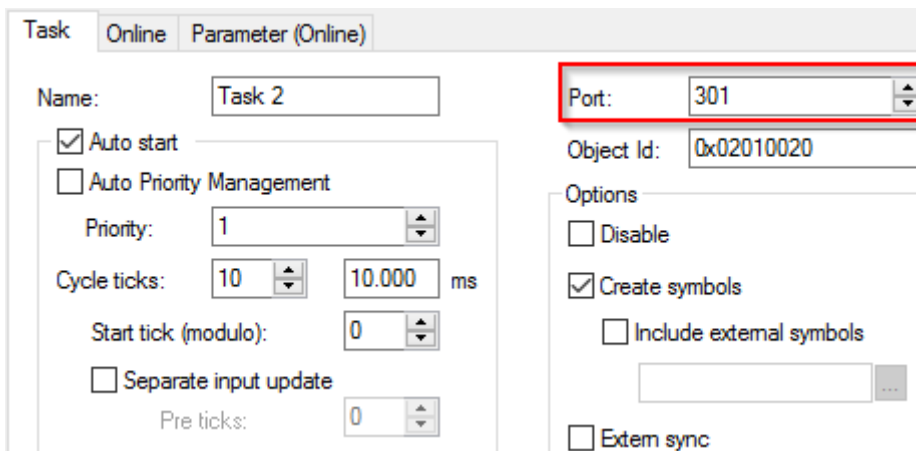
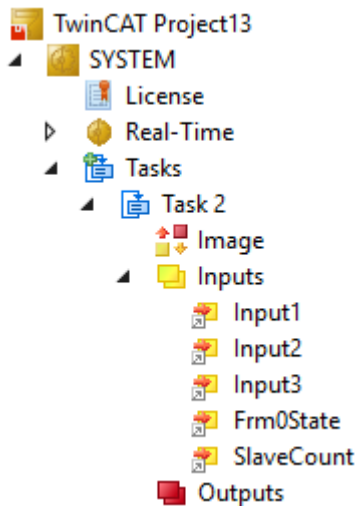


6.2.3 Connect to a TwinCAT I/O Task via ADS

In order to configure the TC3 IoT Data Agent to access a TwinCAT I/O Task (with process image) via ADS, the symbols can be configured manually in the TC3 IoT Data Agent configurator, if the I/O Task has not been configured to create symbol information, or via the target browser.

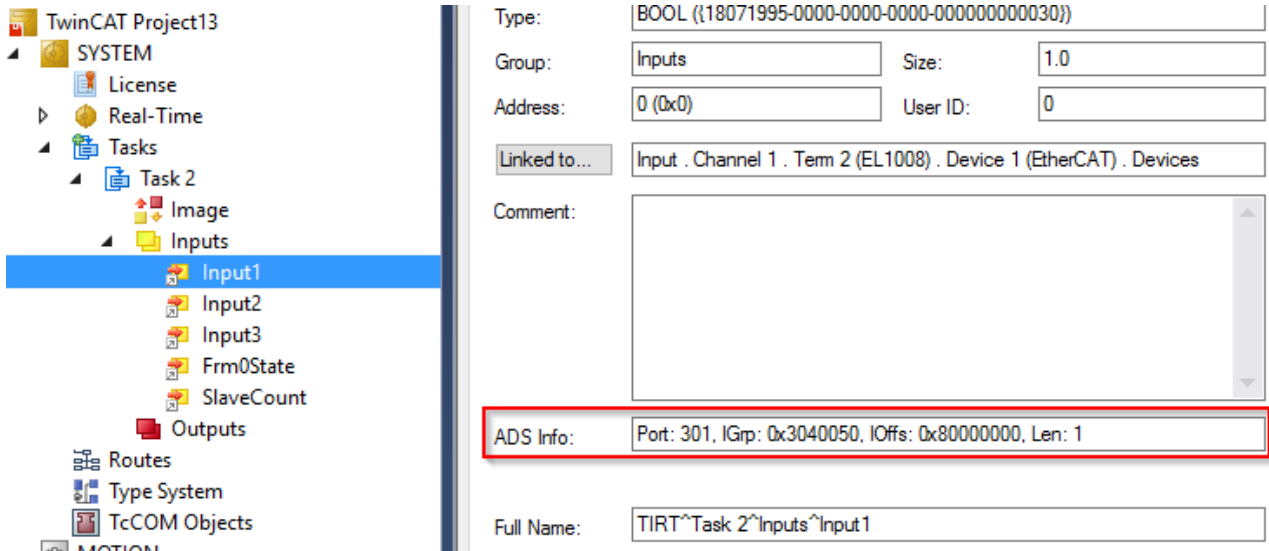
TwinCAT I/O Task

A TwinCAT I/O Task can be configured with a process image in order to link variables from that image directly to I/Os. TC3 IoT Data Agent can then access the I/O Task via its ADS port.



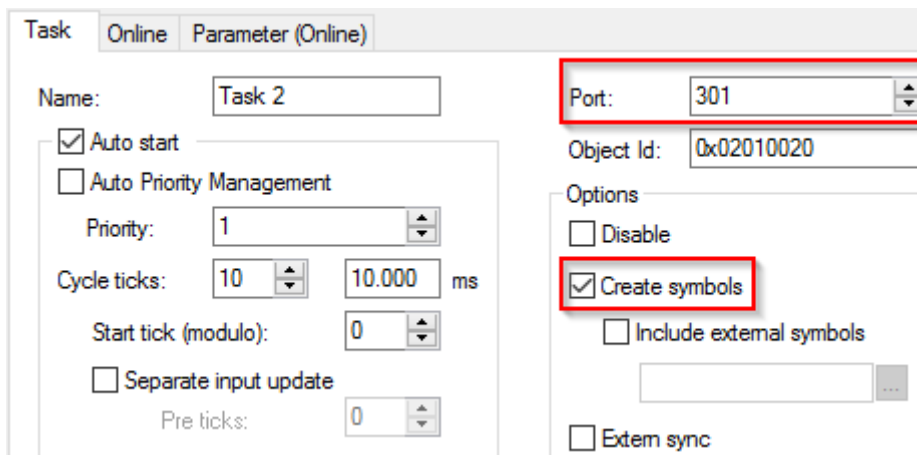
Manual symbol configuration

The process image variables can be accessed via their IndexGroup/IndexOffset combination. This information must be entered into the TC3 IoT Data Agent.

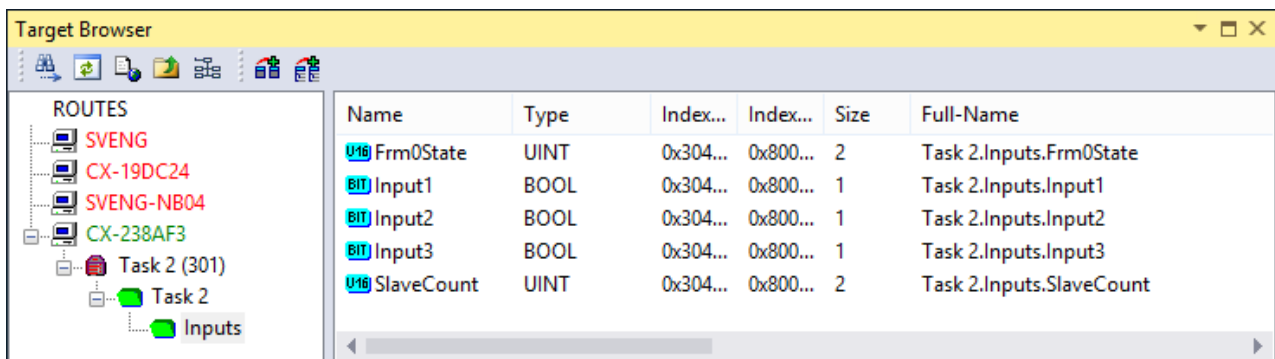


Configuration via target browser

Symbolic information can be generated for a TwinCAT I/O Task by activating the corresponding option in the I/O Task configuration.



After activating this configuration, the symbolic information can be accessed via the target browser and the symbols can be added to the TC3 IoT Data Agent configuration.

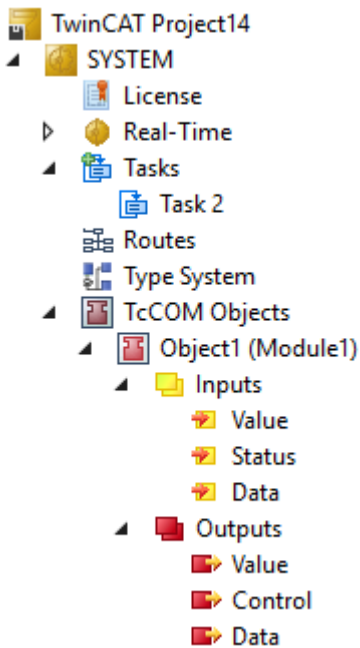


6.2.4 Connect to a TwinCAT TcCOM module via ADS

In order to configure the TC3 IoT Data Agent to access a TwinCAT 3 TcCOM Module via ADS, the symbols can be configured manually in the TC3 IoT Data Agent configurator, if the TcCOM Module has not been configured to create symbol information, or via the target browser.

TwinCAT 3 TcCOM Module

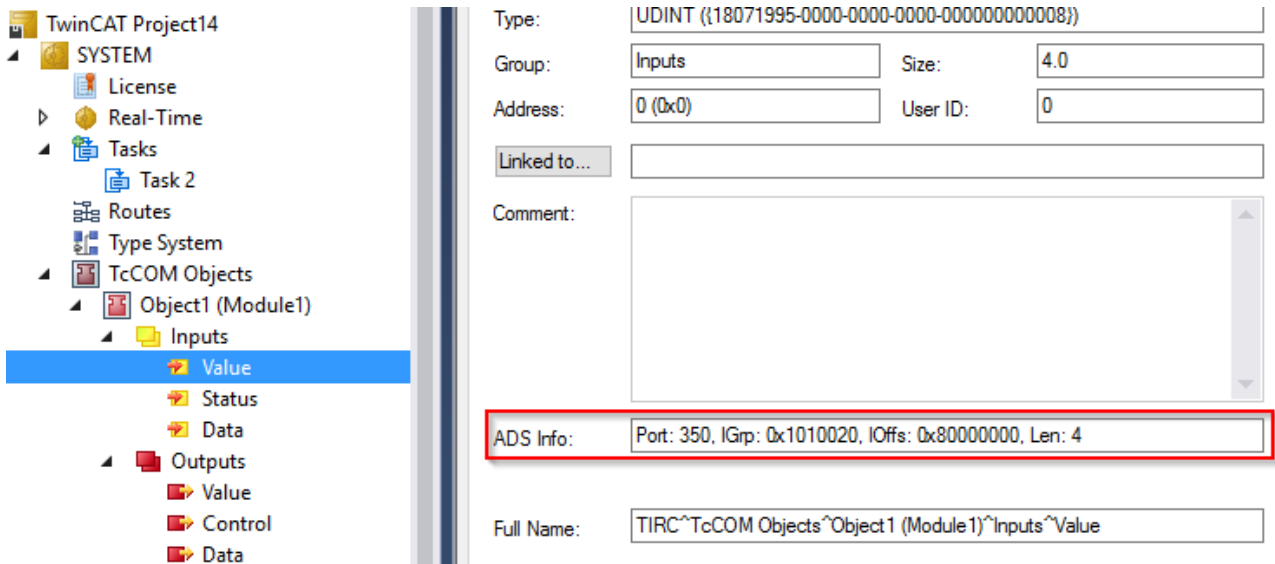
A TwinCAT 3 TcCOM Module can be accessed via ADS via its corresponding ADS port. This is usually the port of the task.



ID	Task	Name	Priority	Cycle Time (µs)	Task Port	Symbol Port
1	02010020	Task 2	1	10000	350	350

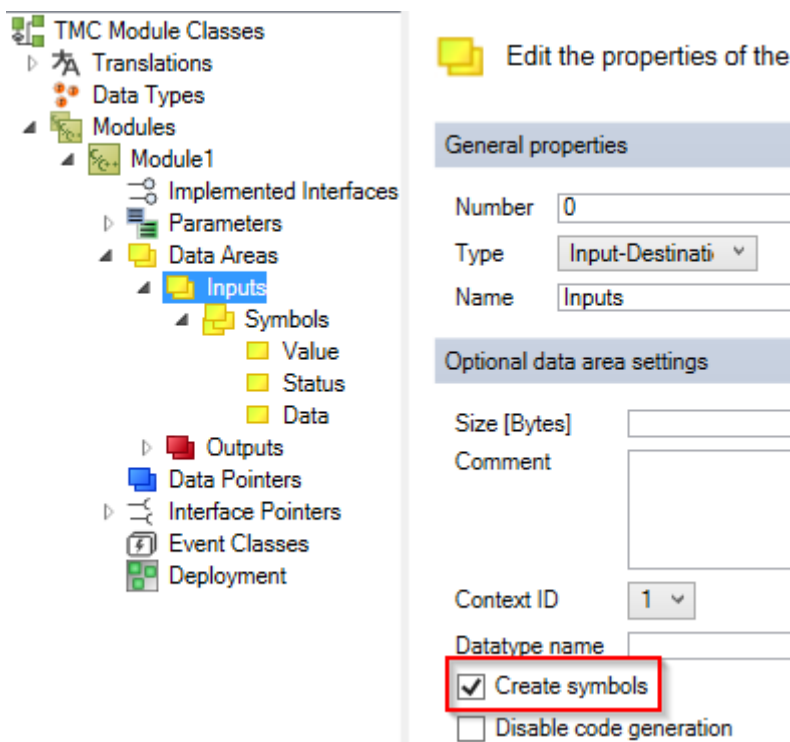
Manual symbol configuration

The process image variables of the TcCOM Module can be accessed via their IndexGroup/IndexOffset combination. This information must be entered into the TC3 IoT Data Agent.



Configuration via target browser

Symbolic information can be generated for a TwinCAT 3 TcCOM Module via the TMC Code Generator.



After activating this configuration, the symbolic information can be accessed via the target browser and the symbols can be added to the TC3 IoT Data Agent configuration.

6.3 Northbound

6.3.1 Publish data to AWS IoT Core

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and to publish it to AWS IoT by using a JSON data format.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
 - 1. Right-click the canvas and select **Add Gate (ADS)**.
 - 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 - 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 - 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 - 5. Open the Target Browser.
 - 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
- ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.



AMS Net ID

Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure AWS IoT gate and channel

Follow the step-by-step instructions to create an AWS IoT gate and a publisher channel.

- ✓ The topology view is activated.
 - 1. Right-click the canvas and select **Add Gate (AWS IoT)**.
 - 2. Select the new gate and configure AWS IoT related settings in the properties window (Broker address, TLS settings).
 - 3. Right-click the new gate and select **Add Channel (Publisher)**.
 - 4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Topic" to "TestTopic". Leave all other fields on their default settings.
- ⇒ An AWS IoT gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically.

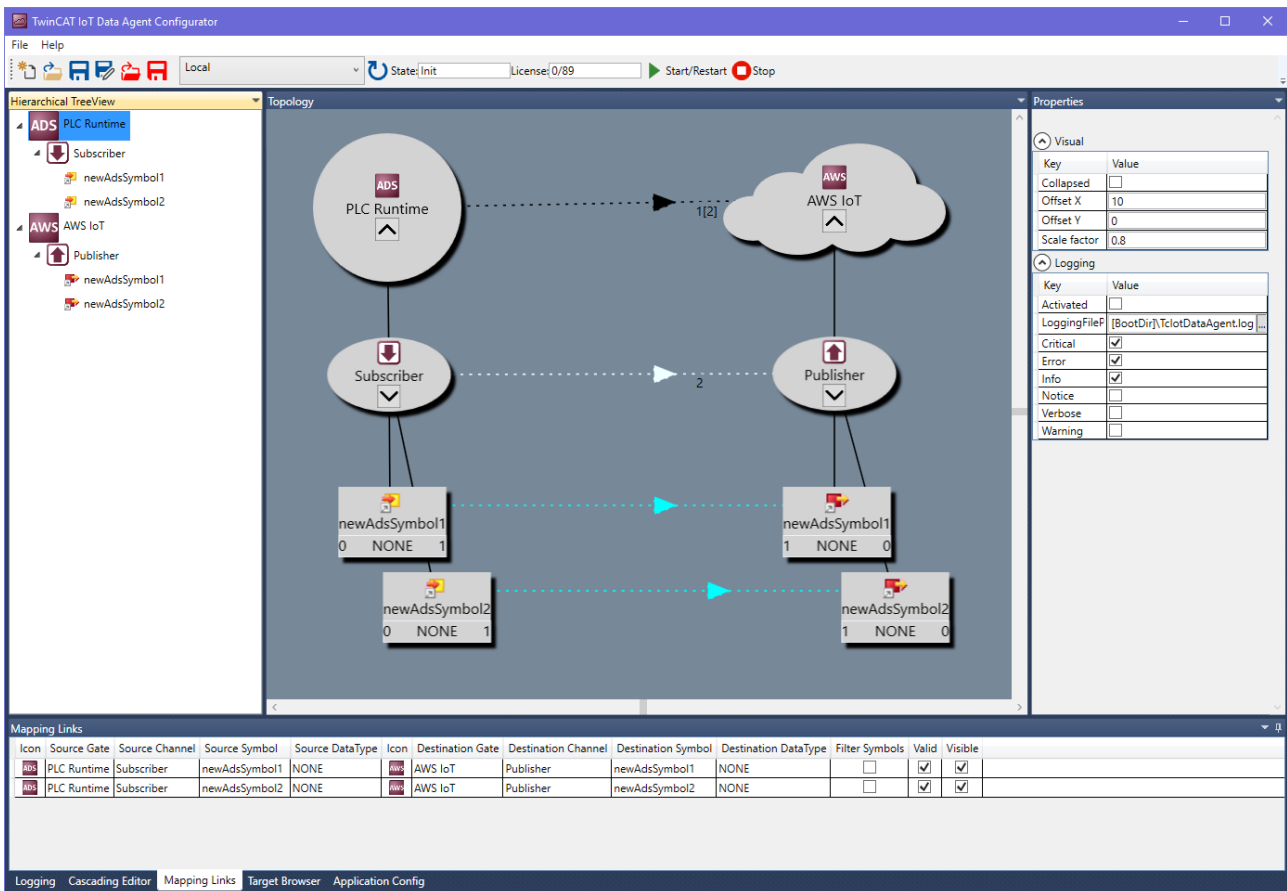
Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.

2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

The resulting configuration looks like this:



6.3.2 Publish data to AWS Greengrass

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and to publish it to AWS Greengrass by using a JSON data format.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
1. Right-click the canvas and select **Add Gate (ADS)**.
 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 5. Open the Target Browser.
 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
 - ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

● AMS Net ID

i Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure AWS IoT gate and channel

A connection to AWS Greengrass can basically be configured like a connection to AWS IoT. Follow the step-by-step instructions to create an AWS Greengrass gate and a publisher channel.

● CA certificate

i Make sure to use the correct CA certificate, which can be obtained via the AWS CLI.

✓ The topology view is activated.

1. Right-click the canvas and select **Add Gate (AWS IoT)**.
 2. Select the new gate and configure AWS Greengrass related settings in the properties window (IP address or hostname of the Greengrass Core device and the TLS settings).
 3. Right-click the new gate and select **Add Channel (Publisher)**.
 4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Topic" to "TestTopic". Make sure to set "QoS" to "0". Leave all other fields on their default settings.
- ⇒ An AWS Greengrass gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

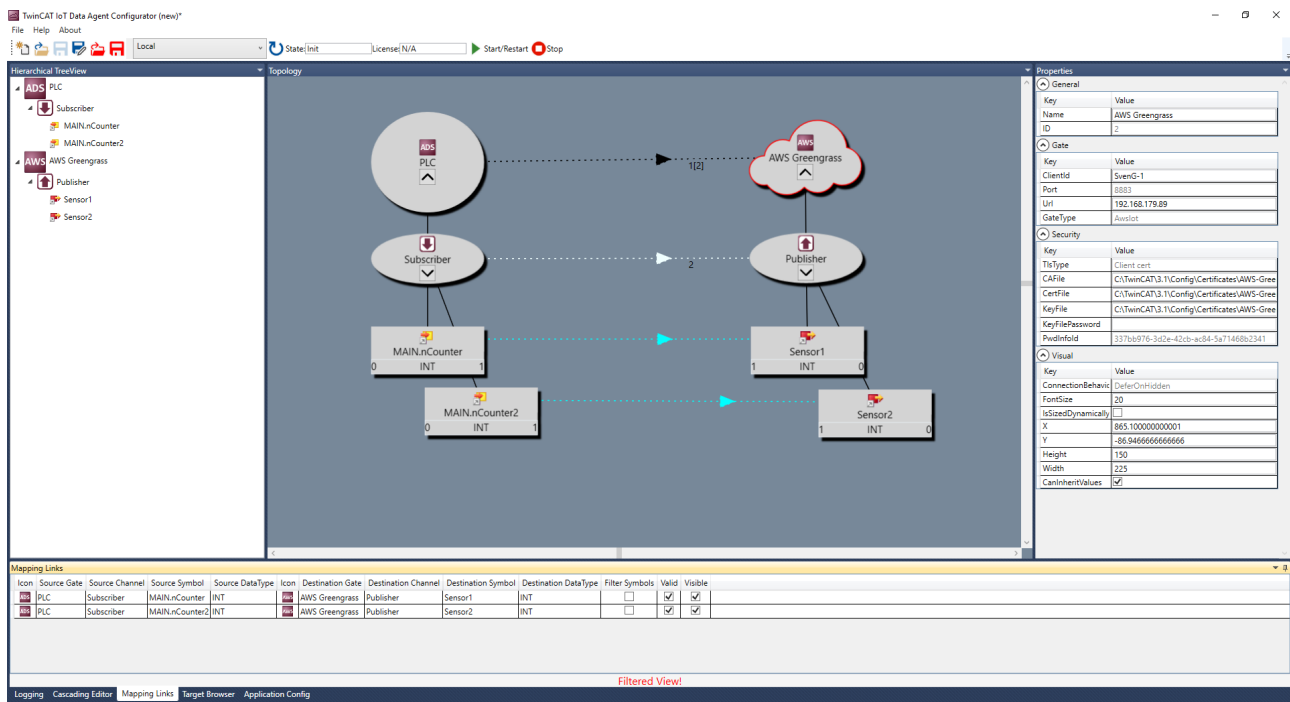
The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
 2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
 3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
- ⇒ The configuration is activated.

The resulting configuration looks like this:



6.3.3 Publish data to IBM Watson IoT

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and publish it to IBM Watson IoT by using a JSON data format.

● IBM Watson IoT settings

i For IBM-specific configuration settings, please consult the IBM Watson IoT documentation. A good Getting Started article can be found here: <https://www.ibm.com/support/knowledgecenter/SSQP8H/iot/platform/devices/mqtt.html>

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
 - 1. Right-click the canvas and select **Add Gate (ADS)**.
 - 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 - 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 - 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 - 5. Open the Target Browser.
 - 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
- ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

● AMS Net ID

i Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure MQTT gate and channel

MQTT is the transport protocol used by IBM Watson IoT. Follow the step-by-step instructions to create an MQTT gate and a publisher channel.

✓ The topology view is activated.

1. Right-click the canvas and select **Add Gate (MQTT)**.
2. Select the new gate and configure IBM Watson IoT related settings in the properties window (Broker address, TLS settings, Username/Password, ClientID) as described in the IBM Watson IoT documentation.
3. Right-click the new gate and select **Add Channel (Publisher)**.
4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Topic" to "iot-2/evt/<eventId>/fmt/<format>". Leave all other fields on their default settings. Replace <eventId> and <format> as described in the IBM Watson IoT documentation.

⇒ An AWS IoT gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

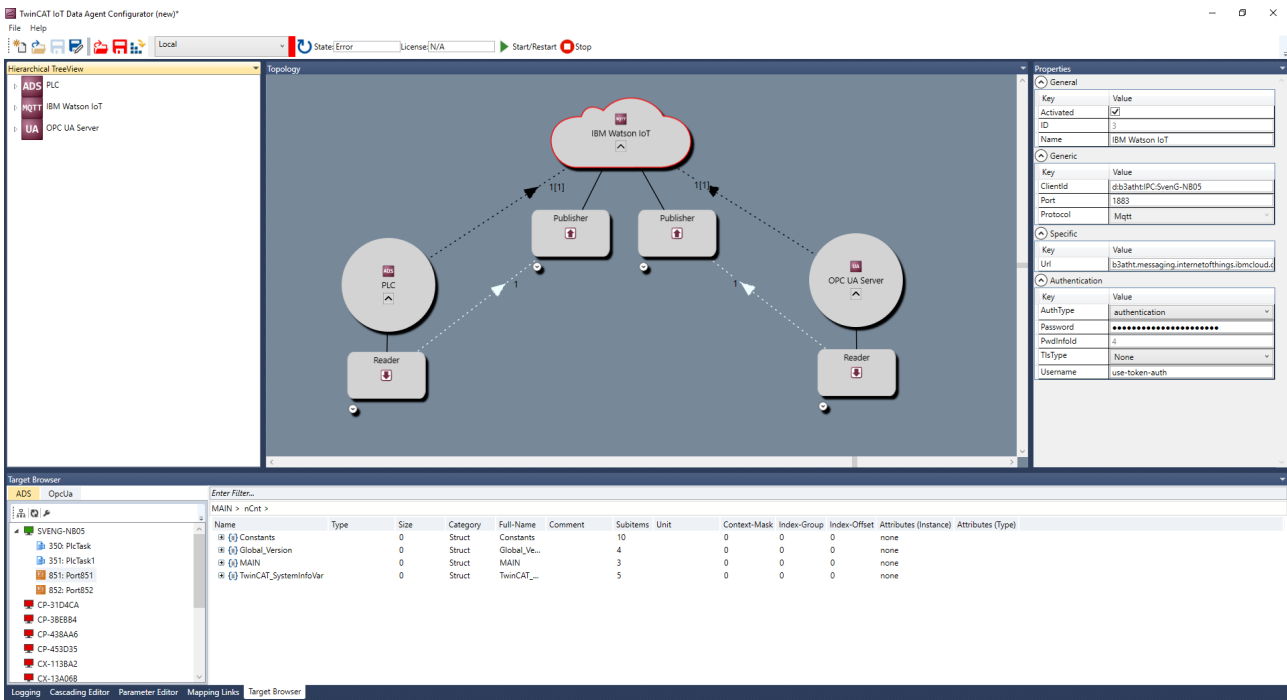
The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

The resulting configuration may look like this:



6.3.4 Publish data to Microsoft Azure IoT Hub

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and to publish it to the Microsoft Azure IoT Hub by using a JSON data format.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
- 1. Right-click the canvas and select **Add Gate (ADS)**.
- 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
- 3. Right-click the new gate and select **Add Channel (Subscriber)**.
- 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
- 5. Open the Target Browser.
- 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.

⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

i AMS Net ID

Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure Azure IoT Hub gate and channel

Follow the step-by-step instructions to create an Azure IoT Hub gate and a publisher channel.

- ✓ The topology view is activated.
- 1. Right-click the canvas and select **Add Gate (IoT Hub)**.
- 2. Select the new gate and configure IoT Hub related settings in the properties window (device name, primary key, CA certificate).

3. Right-click the new gate and select **Add Channel (Publisher)**.
 4. Select the new channel and set “SamplingMode” to “OnChange” and “Formatter” to “Simple JSON”. Leave all other fields on their default settings.
- ⇒ An Azure IoT Hub gate and a publisher channel have been created.

CA certificate



When establishing a connection to the Microsoft Azure IoT Hub, it is required to set a CA certificate. The Baltimore Cyber Trust Root CA can be used as a CA certificate. The associated public key can be extracted from the Microsoft Windows certificate console (Start > Run > mmc.exe, then add the „Certificates“ snap-in). The Baltimore CA can then be found under the heading „Trusted Root Certification Authorities“.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

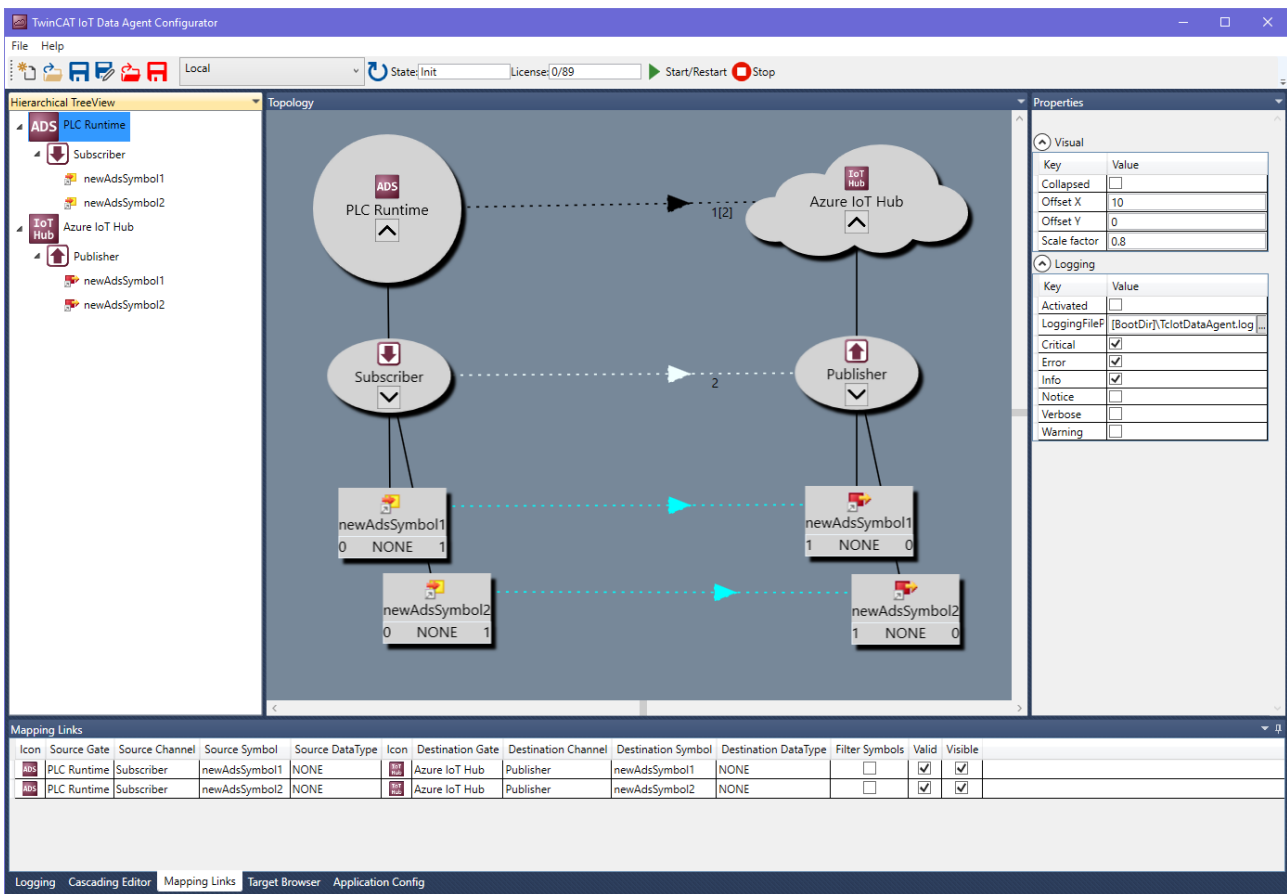
The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

The resulting configuration looks like this:



6.3.5 Publish data to MQTT Message Broker

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and to publish it to a MQTT Message Broker by using a JSON data format.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
 - 1. Right-click the canvas and select **Add Gate (ADS)**.
 - 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 - 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 - 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 - 5. Open the Target Browser.
 - 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
- ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

i AMS Net ID

Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure MQTT gate and channel

Follow the step-by-step instructions to create a MQTT gate and a publisher channel.

✓ The topology view is activated.

1. Right-click the canvas and select **Add Gate (MQTT)**.
 2. Select the new gate and configure MQTT-related settings in the properties window (Broker address, ...).
 3. Right-click the new gate and select **Add Channel (Publisher)**.
 4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Topic" to "TestTopic". Leave all other fields on their default settings.
- ⇒ A MQTT gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

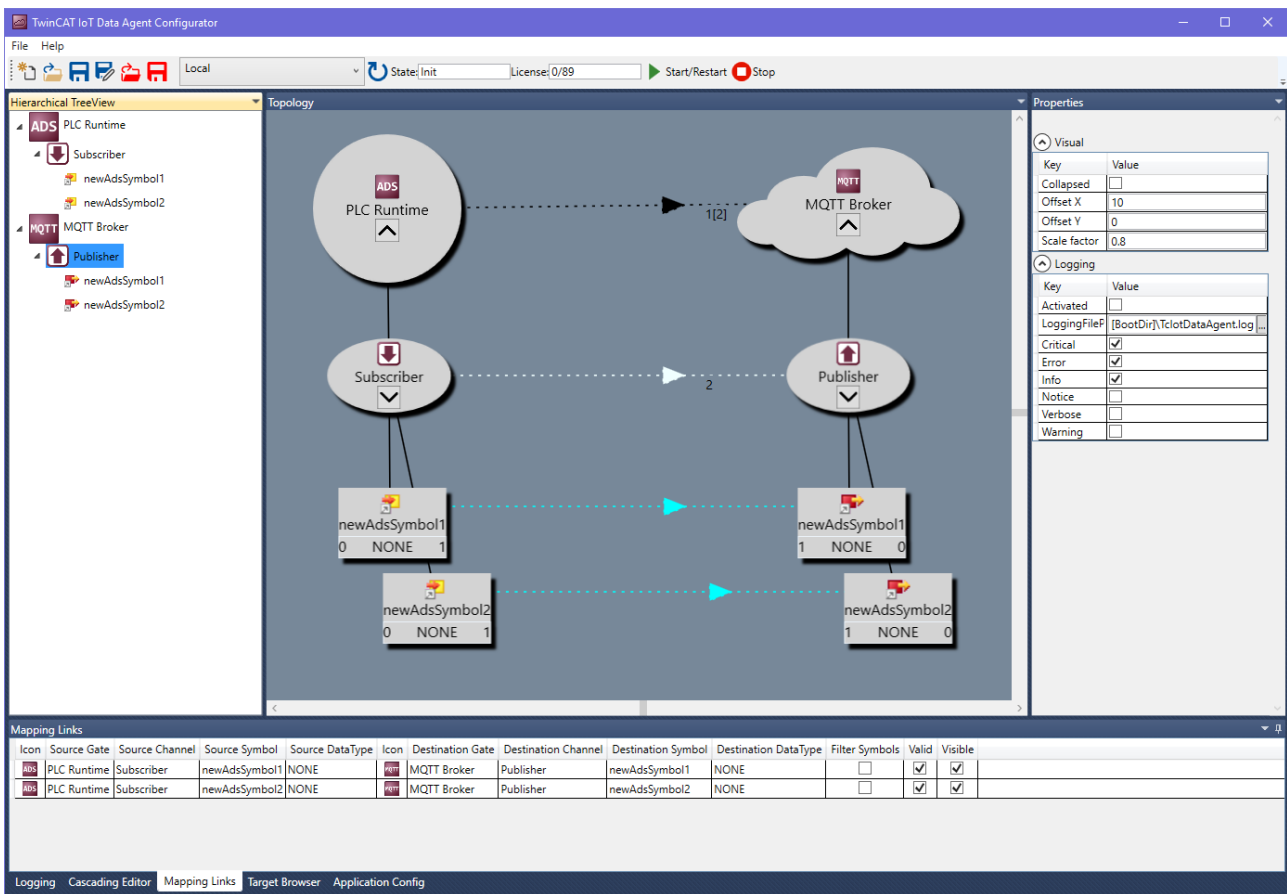
The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
 2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
 3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
- ⇒ The configuration is activated.

The resulting configuration looks like this:



6.3.6 Publish data to TwinCAT IoT Communicator App

The following section describes step-by-step how to use the configurator to sample data from a TwinCAT 3 PLC runtime and publish it to an MQTT Message Broker by using a TwinCAT JSON data format in order to be visualized by the TwinCAT IoT Communicator smartphone app.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
 - 1. Right-click the canvas and select **Add Gate (ADS)**.
 - 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 - 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 - 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 - 5. Open the Target Browser.
 - 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
- ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

i AMS Net ID

Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure MQTT gate and channel

Follow the step-by-step instructions to create a MQTT gate and a publisher channel. The channel is configured so that it represents the topic structure and data format for the TwinCAT IoT Communicator.

1. Right-click the canvas and select **Add Gate (MQTT)**.
 2. Select the new gate and configure MQTT related settings on the new gate (Broker address, ...).
 3. Right-click the new gate and select **Add channel (Publisher)**.
 4. Select the new channel and set "SamplingMode" to "OnChange" and "Formatter" to "TwinCAT JSON". Set "Topic" to <MainTopic>/<DeviceName>/TclotCommunicator/Json/Tx/Data . The parts <MainTopic> and <DeviceName> must be replaced according to your Communicator setup.
 5. Set "StatusInfo Topic" to <MainTopic>/<DeviceName>/TclotCommunicator/Desc. The parts <MainTopic> and <DeviceName> must be replaced according to your Communicator setup.
- ⇒ A MQTT gate and a publisher channel have been created.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

6.3.7 Subscribe to MQTT Message Broker

The following section describes step-by-step how to use the configurator to subscribe to an MQTT Message Broker and to write received data into ADS variables.

Configure MQTT gate, channel and symbols

Follow the step-by-step instructions to create an MQTT gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
1. Right-click the canvas and select **Add Gate (MQTT)**.
 2. Select the new gate and configure MQTT related settings in the properties window (Broker address, ...).
 3. Right-click the new gate and select **Add Channel (Subscriber)**.

4. Select the new channel and set “SamplingMode” to “OnChange” and “Formatter” to “Simple JSON”. Leave all other fields on their default settings. Make sure that the MQTT topic matches the topic name on which you receive data.
 5. Right-click the channel and select **Add Symbol**. Configure the symbol related settings, e.g. the symbol name. In case of a JSON data format, the name must match a JSON key in the JSON document received by the TC3 IoT Data Agent. Make sure to set the correct data type.
- ⇒ A MQTT gate and a subscriber channel have been created. Symbols have been added to the channel.

● Data format



Make sure that the data format of the received message payload corresponds to the configured data format. Otherwise, the TC3 IoT Data Agent cannot parse the content of the message and process the data.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a publisher channel. Add symbols to the channel's configuration.

- ✓ The topology view is activated.
1. Right-click the canvas and select **Add Gate (ADS)**.
 2. Select the new gate and configure ADS related settings in the properties window (AmsNetId, AdsPort, ...).
 3. Right-click the new gate and select **Add Channel (Publisher)**.
 4. Select the new channel and set “SamplingMode” to “OnChange”. Leave all other fields on their default settings
 5. Open the Target Browser.
 6. Navigate to the ADS device and add the symbols to the configuration by dragging them to the publisher channel.
- ⇒ An ADS gate and a publisher channel have been created. Symbols have been added to the channel.

● AMS Net ID



Make sure that the AMS Net ID, that has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.

2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

6.3.8 Use Microsoft Azure IoT Hub Device Twin

The following section describes step-by-step how to use the configurator to set up a connection to Microsoft Azure IoT Hub and the Microsoft Azure IoT Hub Device Twin service.

Configure ADS gate, channel and symbols

Follow the step-by-step instructions to create an ADS gate and a subscriber channel and add symbols to the channel's configuration.

- ✓ The topology view is activated.
1. Right-click the canvas and select **Add Gate (ADS)**.
 2. Select the new gate and configure ADS specific settings in the properties window (AmsNetId, AdsPort, ...).
 3. Right-click the new gate and select **Add Channel (Subscriber)**.
 4. Select the new channel and set "SamplingMode" to "Cyclic". Leave all other fields on their default settings.
 5. Open the Target Browser.
 6. Navigate to the ADS device and add some symbols to the configuration by dragging them to the subscriber channel.
- ⇒ An ADS gate and a subscriber channel have been created. Symbols have been added to the channel.

● AMS Net ID

i Make sure that the AMS Net ID, which has been configured on the ADS gate, matches the ADS device that you selected with the target browser.

Configure Azure IoT Hub gate and Device Twin channel

Follow the step-by-step instructions to create an Azure IoT Hub gate and channels for telemetry data and the Device Twin.

- ✓ The topology view is activated
1. Right-click the canvas and select **Add Gate (IoT Hub)**.
 2. Select the new gate and configure IoT Hub related settings in the properties window (device name, primary key, CA certificate).
 3. Right-click the new gate and select **Add Channel (Publisher)**.
 4. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Name" to "Telemetry Data" (optional). Leave all other fields on their default settings.
 5. Right-click the new gate and select **Add Channel (Device Twin Publisher)**.
 6. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Name" to "Device Twin" (optional). Leave all other fields on their default settings.
 7. Right-click the new gate and select **Add Channel (Device Twin Subscriber)**.
 8. Select the new channel and set "SamplingMode" to "OnChange", "Formatter" to "Simple JSON" and "Name" to "Device Twin GET" (optional). Leave all other fields on their default settings.
- ⇒ You have configured an Azure IoT Hub connection and three channels - one for telemetry data and two for the Device Twin. In the step you have to decide which source symbols from the ADS gate will be treated as telemetry data and which symbols should be send to the Device Twin via the Device Twin Publisher channel. The Device Twin Subscriber channel can be used to retrieve updates to the Device Twin from cloud-based backend applications (so-called "desired" updates). Every property expected has to be added as a symbol to that channel, using the property name as symbol name and URN.

Create mapping

Follow the step-by-step instructions to create a mapping between source and destination symbols. Note that a mapping can only be created between channels that have different roles (e.g. between a subscriber and a publisher channel and vice versa).

Drag-and-drop - Channel:

While holding the CTRL key, drag the subscriber channel to the publisher channel.

The configurator automatically creates new symbols on the publisher channel and a mapping between the source and destination symbol.

Drag-and-drop – Symbol:

While holding the CTRL key, drag a symbol from the subscriber channel to either an existing symbol on the publisher channel or to the publisher channel itself.

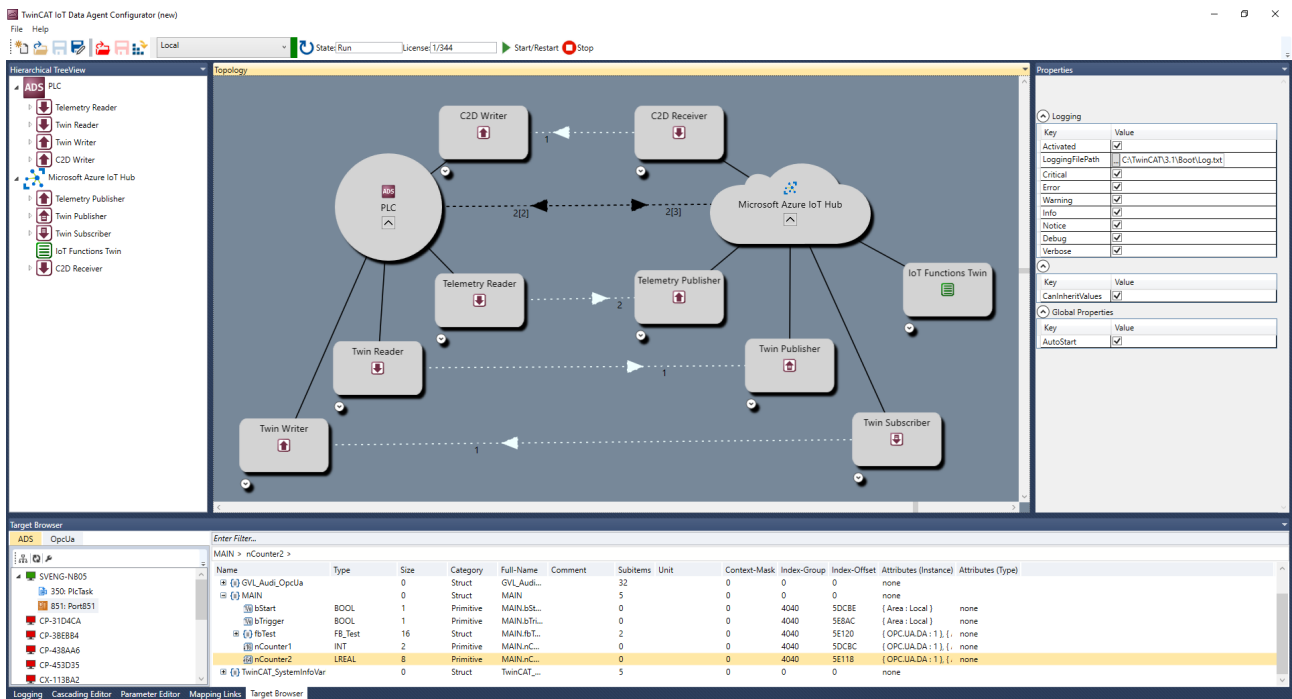
The configurator creates a link between source and destination symbol or, if the symbol has been dragged to a channel object, creates the destination symbol automatically

Activate configuration

Follow the step-by-step instructions to activate the currently opened configuration.

1. Select the target system via the toolbar.
2. Click the **Activate Configuration** button.
 - ⇒ A dialog opens asking if you want to switch the target TC3 IoT Data Agent into run mode.
3. Confirm the dialog with **Yes** if you want to switch the mode. Otherwise, choose **No**.
 - ⇒ The configuration is activated.

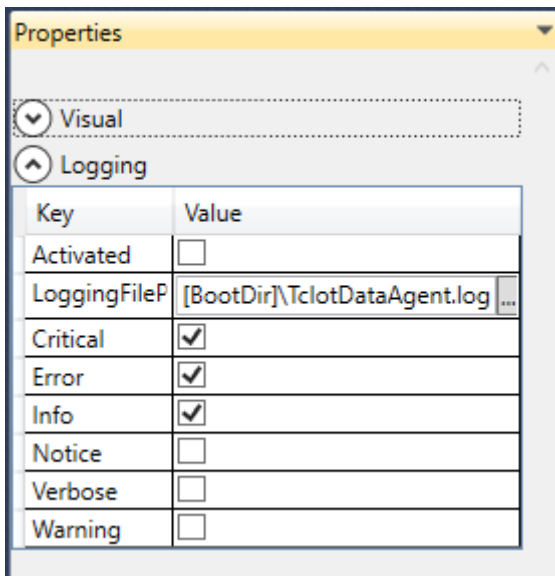
The resulting configuration may look like this:



7 Appendix

7.1 Error logging

For troubleshooting purposes, the TC3 IoT Data Agent can generate a logfile, which can be populated based on different logging levels. All necessary settings can be configured via the properties window of the configurator. Simply click on an empty spot on the topology view and configure the logging settings in the properties window.



Note that all settings are tied to the currently opened configuration and have to be set individually for every configuration. The default directory in which the logfile is created, is the TwinCAT boot directory. The placeholder [BootDir] automatically selects the TwinCAT boot directory.

i Logging window

The logging window inside of the configurator is only showing log events that are related to the configurator. In order to create a log for the TC3 IoT Data Agent background service, the settings above are required.

7.2 Error diagnosis

This documentation article explains some common configuration or setup mistakes and might provide a good way to troubleshoot your setup. It might also help to create an [error logfile](#) [▶ 74].

Behavior	Description
System tray icon turns RED after start	<p>Double-check all settings and turn on logging [► 74]. In addition check the following things:</p> <ul style="list-style-type: none"> • Is the TC3 IoT Data Agent able to connect to all configured gates (see below)? • Are there enough licenses available?
Connection to MQTT Message Broker could not be established	<p>Check the following things:</p> <ul style="list-style-type: none"> • Make sure that the Message Broker is reachable for the device that executes TC3 IoT Data Agent. The default MQTT port for an unencrypted MQTT communication is 1883/tcp and 8883/tcp if MQTT-TLS is used. Make sure that these ports are opened in your firewall (on the Message Broker side for incoming access and on the TC3 IoT Data Agent side for outgoing access). • Make sure that you use the correct access credentials (if applicable) for the message broker • Check connectivity with the message broker by using another MQTT client, e.g. Mqtt.Fx, MqttSpy or the Mosquitto command line tools.
Connection to ADS device could not be established	<p>Check the following things:</p> <ul style="list-style-type: none"> • Make sure that you use the correct ADS port and AmsNetID for the remote device • Make sure that the remote device does not have any firewall settings in place that prohibit ADS access. • Make sure that the computer that executes the TC3 IoT Data Agent has an ADS route configured to the corresponding TwinCAT target device. (Only applicable if the TC3 IoT Data Agent and ADS target runtime are on different computers)
Connection to OPC UA Server could not be established	<p>Check the following things:</p> <ul style="list-style-type: none"> • Make sure that the computer that executes the OPC UA Server does not have any firewall settings in place that prohibit remote access to the server. • Check any local firewall settings on the TC3 IoT Data Agent device.
Connection to Azure IoT Hub could not be established	<p>Check the following things:</p> <ul style="list-style-type: none"> • Make sure that you use the correct IoT Hub hostname • Make sure that you have created a device on Azure IoT Hub, e.g. by using the Azure Device Explorer, and that you have entered the correct device credentials for that device in the TC3 IoT Data Agent configurator • Make sure that you have used the correct CA certificate for Azure IoT Hub and that the CA certificate is used in a Base64-encoded file format. • Make sure that you do not use the same access credentials on another device. Only one connection can be established to IoT Hub with the same set of access credentials.
Connection to AWS IoT could not be established	<p>Check the following things:</p> <ul style="list-style-type: none"> • Check connectivity with AWS IoT by using another MQTT client, e.g. Mqtt.Fx, MqttSpy or the Mosquitto command line tools. • Make sure that no firewalls are blocking the TCP port 8883 (outgoing), which is required for connectivity with AWS IoT. • Make sure that you have used the correct set of certificates (as supplied by the AWS IoT management website)

7.3 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.

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:

<http://www.beckhoff.com>

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49(0)5246/963-0
Fax:	+49(0)5246/963-198
e-mail:	info@beckhoff.com

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(0)5246/963-157
Fax:	+49(0)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(0)5246/963-460
Fax:	+49(0)5246/963-479
e-mail:	service@beckhoff.com

More Information:
www.beckhoff.com/tf6720

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

