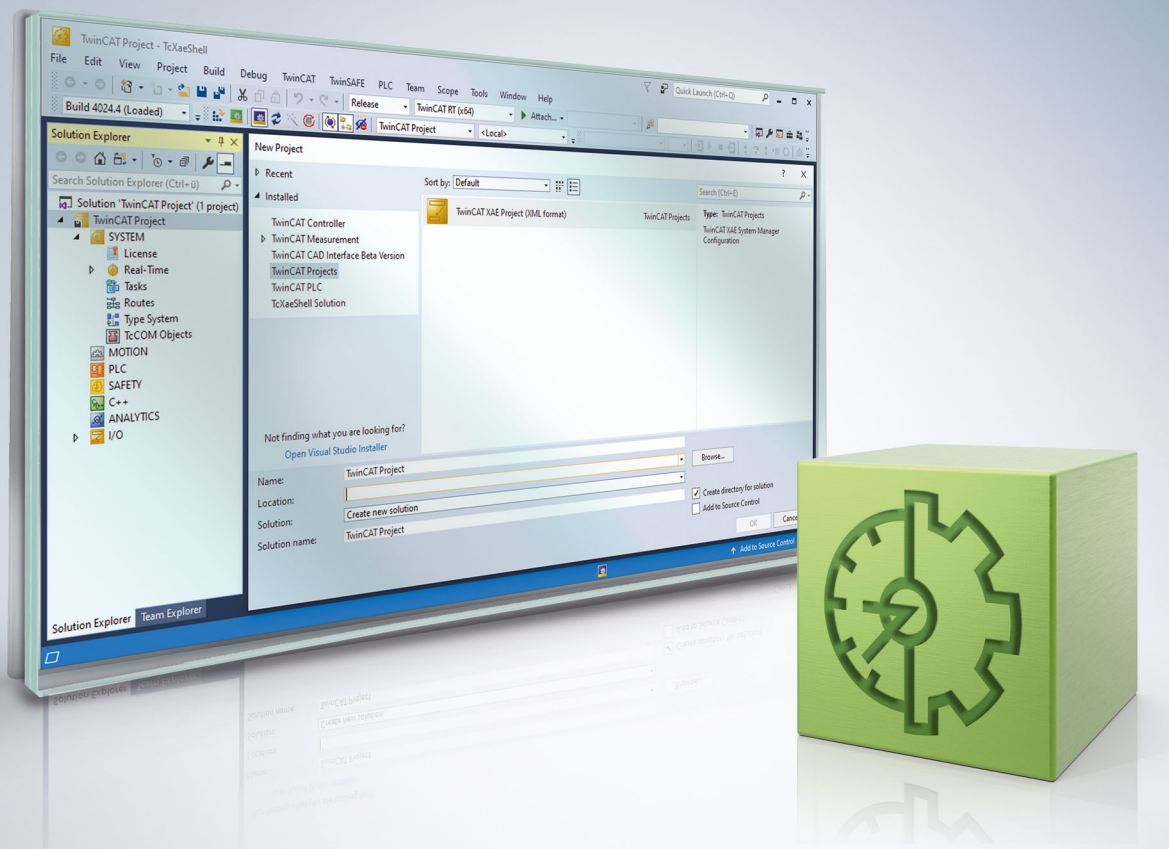
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | ADS-over-MQTT



2025-01-30 | Version: 1.4.0

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

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

Patents

The EtherCAT Technology is covered by the following patent applications and patents, without this constituting an exhaustive list:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

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

Copyright

Third-party brands

Third-party trademarks and wordmarks are used in this documentation. The trademark endorsements can be found at: https://www.beckhoff.com/trademarks

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

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

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

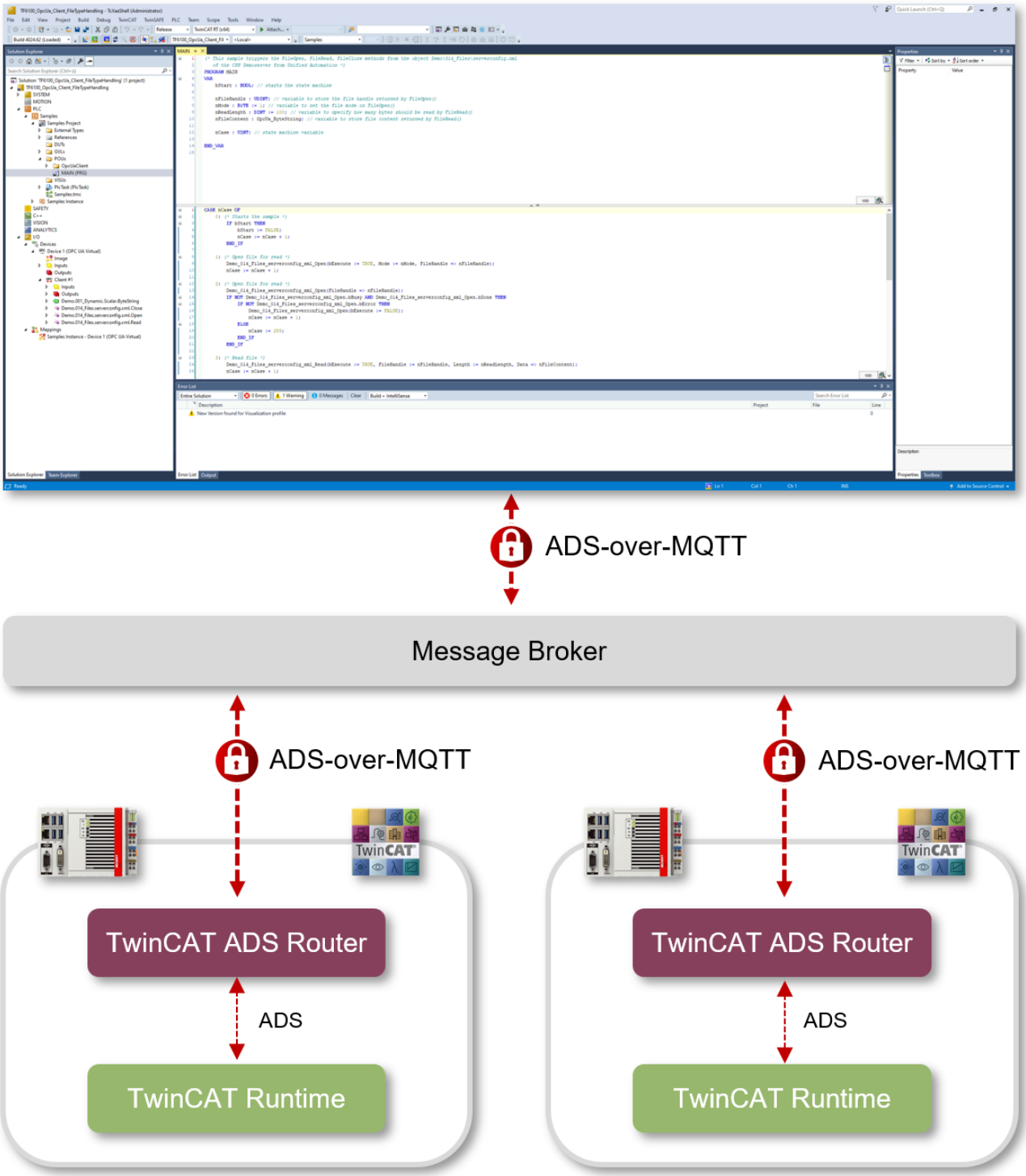
2 Overview

Beckhoff ADS (Automation Device Specification) is a communication protocol developed by Beckhoff for efficient data exchange in industrial automation systems. It serves as the backbone for the integration of devices and software into the PC-based control technology from Beckhoff.

From the perspective of the ADS protocol, ADS-over-MQTT is an additional transport channel over which ADS can be transported. Decoupling communication via an MQTT message broker results in a number of advantages, particularly in terms of scalability and flexibility when integrating additional ADS applications. Security mechanisms such as TLS can be used at the transport layer to secure the communication connection.

With ADS-over-MQTT, the entire data exchange is transparent for the ADS applications, because only the ADS router needs to know and hold the corresponding information on the MQTT transport channel. In particular, this also enables easy retrofitting for existing applications.

The main use case for ADS-over-MQTT is a classic remote maintenance and remote diagnostics scenario, where the TwinCAT engineering environment (TwinCAT XAE) needs to connect to one or more controllers for remote debugging. The following diagram illustrates the architecture being created here.



However, there are many other use cases for ADS-over-MQTT, especially when it comes to the aggregation of multiple distributed PLC systems.

This document provides an overview of the usage possibilities as well as a technical description of how a "virtual ADS network" can be configured over an MQTT message broker.

Benefits of an MQTT-based ADS network

- **Subnets, NAT-based networks and firewalls:**
  Incoming TCP/IP connections are used in both directions in a classic ADS setup. This makes it necessary for the devices to be located in the same network in the normal case. In distributed systems with different subnets this leads to complex configurations in order to make the ADS routes available. In the case of MQTT-based ADS networks, only an outgoing TCP/IP connection is used by the devices. This allows the broker in the higher-level network to broker between all devices. Due to the outgoing connections, a typical firewall can be used and no incoming ports need to be registered.

- **Access control:**
  After creating the appropriate routes, bidirectional communication can be executed in a classic ADS setup. An access by device A, which accesses B, also allows device B to access A. The MQTT-based ADS network can be configured so that device A can access B, but not the other way around.
- **Security / encryption:**
  The communication from TwinCAT to the broker can be encrypted by TLS (with certificates or PreSharedKey (PSK)). In this case, the transporting MQTT protocol is encrypted, so the ADS protocol can be transmitted unencrypted in the payload.
- **Retrofitting:**
  ADS-over-MQTT is transparent for the ADS applications, which means that they do not need to be changed.

NOTICE

ADS access means full access

As described in Security Advisory 2017-01, ADS offers full access to a device.
Secure ADS offers authorization as well as encryption for the communication; therefore, it represents a transport encryption. Hence, if an ADS route exists, then full access exists.
Dedicated, role-related access to individual files is offered by solutions such as OPC UA.

3 Installation

3.1 System Requirements

The following system requirements apply for the installation and operation of this product.

Technical data	Description
Operating system	Windows 10 Windows Server 2022 TwinCAT/BSD TwinCAT/Linux®
Target platforms	PC architecture (x86, x64, Arm®)
TwinCAT version	TwinCAT 3 (from Build 4022.0)
TwinCAT installation level	TwinCAT 3 XAE, XAR, ADS
Required TwinCAT license	---

MQTT Message Broker

To use this product, an MQTT message broker is required via which the communication connection is established. Any MQTT message broker can be used, e.g. Mosquitto or HiveMQ. The use of managed cloud services, such as AWS IoT Core, is also possible.

Plugin for Mosquitto Message Broker

The supplied plugin for the Mosquitto Message Broker is currently only available for the Windows operating system.

3.2 Installation

The ADS-over-MQTT feature is a fixed part of the basic TwinCAT installation, both on XAE, XAR and ADS installations, and no further installations are required on the TwinCAT side. To install the MQTT message broker, please consult the documentation for your message broker software.

4 Technical introduction

4.1 Quick Start

This documentation article is intended to allow you an initial, quick start in how to use this product. Carry out the following steps to establish an ADS-over-MQTT connection to an MQTT message broker and to send and receive ADS messages.

> **Message Broker installation**
>
> This document assumes that you have a locally installed and working MQTT message broker. As an example we use the Mosquitto Message Broker here, but you can use any message broker.

Overview

For demonstration purposes, two TwinCAT devices are to establish an ADS-over-MQTT connection with the message broker. We then use TwinCAT XAE (Engineering) on the first device to establish a connection to the second system via the ADS-over-MQTT route. To keep the installation scenario as simple as possible, both devices should be on the same network.

Device 1:

- Mosquitto Message Broker
- TwinCAT 3 XAE

Device 2:

- TwinCAT 3 XAE or XAR

Note the IP address or the host name of Device 1. In the latter case, make sure that the name resolution works in your network.

Message Broker setup

The Mosquitto Message Broker has been delivered since version 2.x with a configuration that requires security measures to be set up to ensure secure operation of the message broker. In the following, we will show you how to modify the Mosquitto Message Broker configuration so that an unsecured communication connection can be established with the broker. However, this should be done exclusively for testing purposes in a trusted operating environment. For productive operation, we recommend using a secure broker configuration.

1. Install the Mosquitto Message Broker on your system.
2. Make a backup of the *mosquitto.conf* file from the Mosquitto installation directory. This is typically located at *C:\Program Files\mosquitto.*
3. Open the *mosquitto.conf* file with a text editor of your choice and remove the existing content. Add the following content and save the file.

```
listener 1883
allow_anonymous true
```

4. Restart the Mosquitto Message Broker, either via the corresponding Windows service or manually via the console or *mosquitto.exe*.

⇨ You have now configured the Mosquitto Message Broker to listen for incoming client connections on port 1883 and it does not require any security (neither user authentication nor client certificates).

> **Windows firewall on Device 1**
>
> Please enable the standard MQTT port 1883/tcp as an incoming port in the Windows firewall of Device 1 so that Device 2 can establish a connection to the message broker.

You can now start configuring TwinCAT.

Configuration of the first TwinCAT device

The TwinCAT ADS router on this device should now establish a route to the local message broker. Create a new XML file, e.g. with Notepad, and insert the following content.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="1883">127.0.0.1</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

Save this XML file under any name in the following directory and restart TwinCAT for the changes to take effect.

```
\TwinCAT\3.1\Target\Routes
```

Configuration of the second TwinCAT device

The TwinCAT ADS router on this device should establish a route to the message broker on device 1. Create a new XML file, e.g. with Notepad, and insert the following content.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="1883">%IPAddress%</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

Replace the entry %IPAddress% with the IP address or the host name of device 1. Save this XML file under any name in the following directory and restart TwinCAT for the changes to take effect.

```
\TwinCAT\3.1\Target\Routes
```

Connecting the engineering

Now start TwinCAT XAE on Device 1 and create a new TwinCAT project. Alternatively, you can also open an existing project. For this tutorial, we only need the toolbar to establish a connection to a remote ADS device.

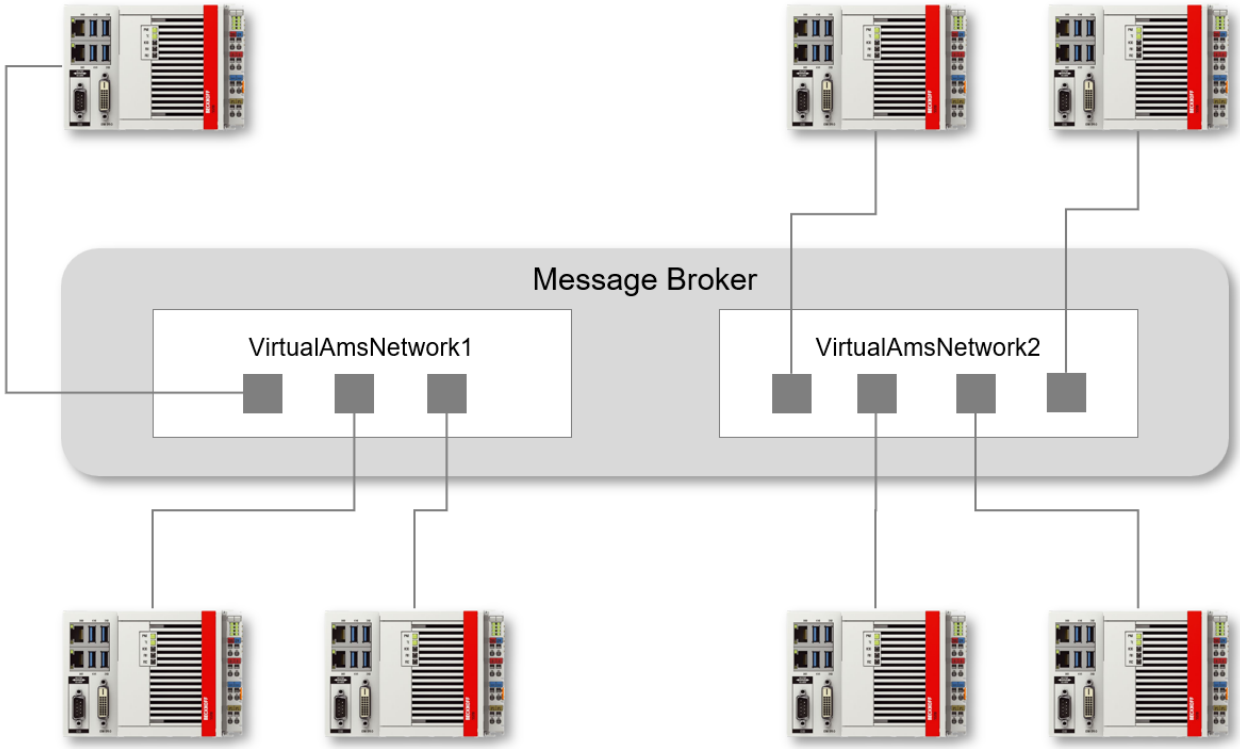
You will now find TwinCAT Device 2 in the toolbar and can establish a connection to it.

Next steps

After you have successfully established an ADS-over-MQTT connection to the message broker, we recommend that you reset the Mosquitto Message Broker to its default settings. For this, please take the backup file of *mosquitto.conf* that you created in the previous steps. This file, together with the broker documentation, is a good basis for further steps, e.g. to set up a secure message broker operating environment considering client/server certificates and user authentication.

4.2 Virtual networks

When configuring ADS-over-MQTT, so-called "virtual networks" can be defined. This allows ADS devices to be distributed into independent networks. Only the devices within a network can communicate with each other. At MQTT level, this distribution takes place on the basis of a common basic topic.



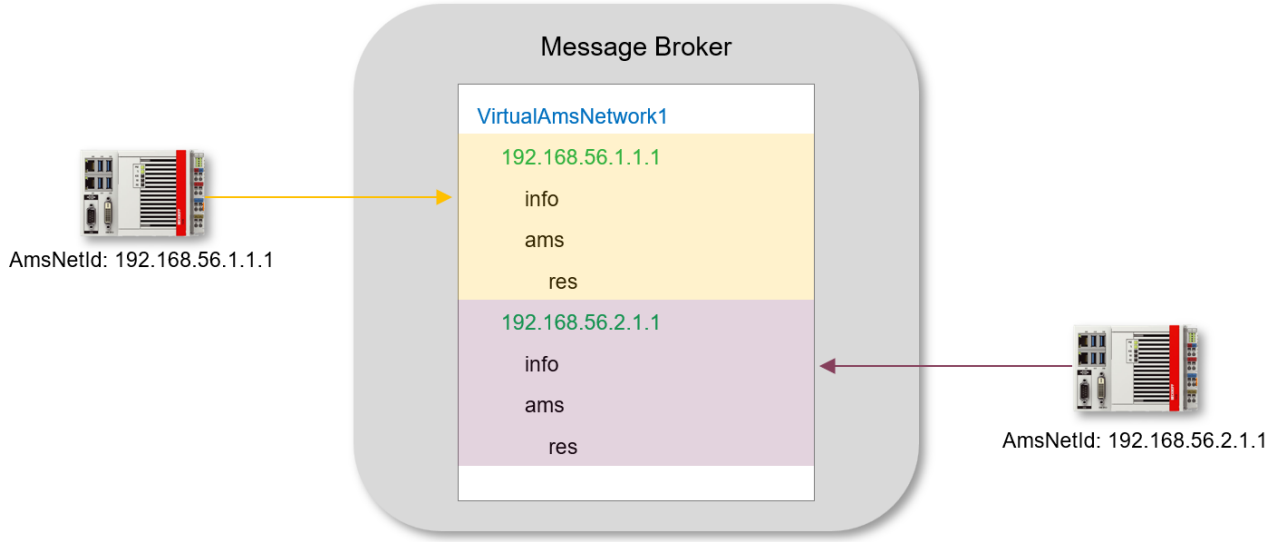
These virtual networks can be defined at client level using the ADS-over-MQTT configuration file [▶ 16] and, if required, access rights can be assigned using the TcMqttPlugin [▶ 18] for the Mosquitto Message Broker.

4.3 Communication flow

To enable data exchange and Device Discovery via ADS-over-MQTT, all ADS devices use a uniform topic structure on the message broker. The topic structure depends on the name of the configured virtual network [▶ 13] and the AMS Net ID of the respective device.

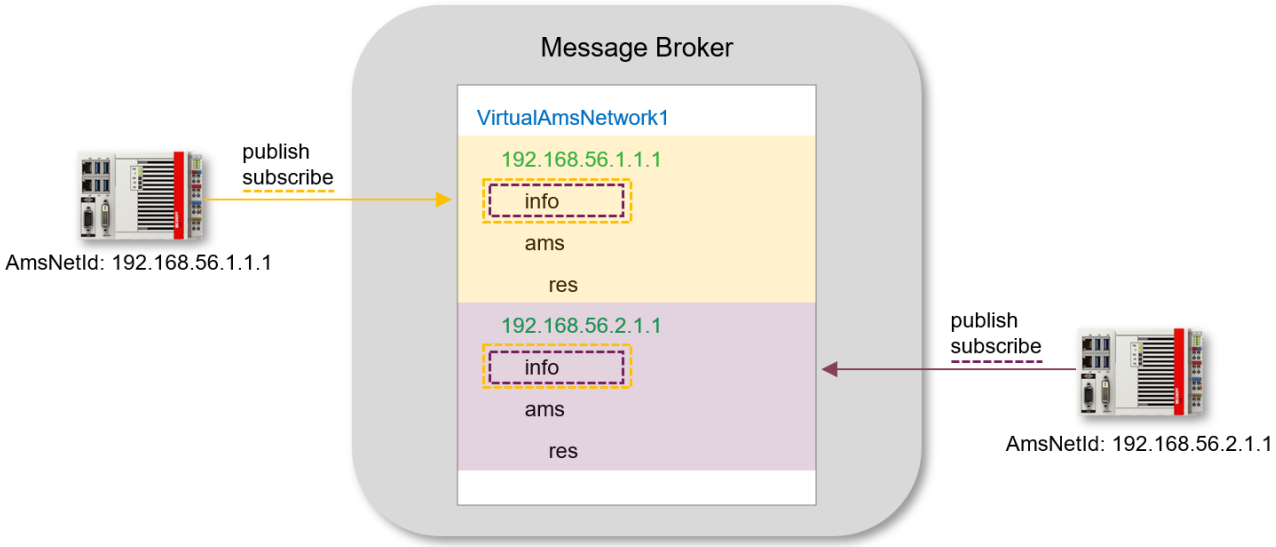
Type	Topic
Discovery	<NetworkName>/<AmsNetId>/info
Communication	<NetworkName>/<AmsNetId>/ams <NetworkName>/<AmsNetId>/ams/res

Each ADS device therefore has its own "topic area" on the message broker. The following figure illustrates this relationship.



Discovery

A connecting TwinCAT ADS router sends a retain message with device information to its discovery topic, at the same time it subscribes to the topic <NetworkName>/+/info, so that it is informed about all other connected routers.



The messages to the discovery topic contain an XML structure with device information, for example the host name and the TwinCAT version used:

```
<info>
  <online name="EC2AMAZ-2RRSQS6" osVersion="10.0.20348" osPlatform="2"
tcVersion="3.1.4026.10">true</online>
</info>
```

If the message broker does not support retain messages, this can be taken into account in the ADS-over-MQTT configuration file [▶ 16]. In this case, a communication handshake would take place instead of a retain message: a newly connecting device logs on to its discovery topic and all other connected devices respond with another message on their discovery topic. This ensures that devices can find each other even with message brokers that do not support retain messages. The disadvantage is an increased volume of messages in larger operating environments with frequent reconnects.

Communication

A TwinCAT ADS router subscribes to its communication topic (<NetworkName>/<AmsNetId>/ams/#) immediately after the connection is established. The ADS commands to this router are then sent to <NetworkName>/<AmsNetId>/ams, while the responses are received via the <NetworkName>/<AmsNetId>/ams/res topic.

4.4 Configuration file

The TwinCAT ADS router is configured by an XML file to establish an ADS-over-MQTT connection with a message broker. This configuration file is stored under any name in the following directory:

```
\TwinCAT\3.1\Target\Routes
```

In the Samples [▶ 21] chapter, you will find sample configuration files for all the use cases described below.

Activating a new ADS-over-MQTT configuration

New or changed ADS-over-MQTT configurations are only adopted when the TwinCAT ADS router is initialized. This takes place, for example, in the TwinCAT transitions from RUN to CONFIG or CONFIG to CONFIG.

Path information

Please ensure that you use the correct spelling for your operating system for any path details in the configuration file.

Basic configuration

The basic configuration always contains the following elements:
Address of the message broker, its TCP port and the name of the virtual network [▶ 13].
The address of the broker and the TCP port at which it can be reached are specified via the <Address> node. The virtual network is defined via the <Topic> node. In the following example, a connection is established to a locally (127.0.0.1) installed message broker and TCP port 1883. "VirtualAmsNetwork1" is defined as the virtual network".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
 <RemoteConnections>
  <Mqtt>
   <Address Port="1883">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

NoRetain

The communication flow [▶ 14] for device discovery can be customized via the NoRetain attribute. By setting NoRetain = true, the device search function is no longer based on retain messages. Instead, a handshake mechanism is used to identify all connected ADS devices.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt NoRetain="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

Unidirectional

Incoming ADS messages can be blocked for this system via the unidirectional attribute. A typical use case could be an engineering system, for example, which should be able to reach the runtime systems via ADS, but not vice versa.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt Unidirectional="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS

The <TLS> node can be used to define settings for securing the transport channel via TLS. Various connection options are available, e.g. the configuration of client certificates or PSK. Our samples [▶ 21] show all possible configuration variants.

TLS IgnoreCn

The IgnoreCn attribute can be used to disable the verification of the CommonName (CN) from the server certificate.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt>
        <Address Port="8883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Tls IgnoreCn="false">
            <Ca>C:\TwinCAT\3.1\Target\Certificates\mybroker\CA.pem</Ca>
        </Tls>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS PSK

When using TLS with a Pre-Shared Key (PSK), you can either specify the PSK as a hex-coded, 64-character string, or leave the conversion to TwinCAT internally using Sha256(Identity + Pwd). In the latter case, the IdentityCaseSensitive attribute can be used to specify that TwinCAT should use the identity as the UpperCase for the calculation. Our samples [▶ 21] show both possible configuration variants.

User

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
 <RemoteConnections>
  <Mqtt>
   <Address Port="1883">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
   <User>CX-12345</User>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

The <User> element specifies an identity that can be used in the TcMqttPlugin [▶ 18] to configure access rights between ADS devices. However, the identity is usually defined by other means, e.g. the CommonName (CN) of a client certificate.

Optionally, the <Mqtt> element can contain a ClientId attribute to specify the MQTT ClientID. This is otherwise formed from the <User> and an arbitrary string.

4.5 Security

There are options for securing the communication. A TLS connection on the basis of X.509 certificates or a Pre-Shared Key (PSK) can be used for this. It is recommended that communication be secured with TLS especially when communicating over non-trustworthy networks (e.g. the Internet). In the chapters Configuration file [▶ 16] and Samples [▶ 21] you will find explanations and sample configuration files for operating ADS-over-MQTT via TLS.

The broker itself must be operated in a trustworthy environment, as all messages on the broker are unsecured.

Compromising of the virtual ADS network

Even when communication between the devices and the broker takes place in encrypted form via TLS, the devices are not secured among one another. The ADS commands are present on the broker in unencrypted form.
If a device is compromised, the attacker can execute all ADS commands via the rights gained. These commands also include file reading operations or operations for starting processes.

Two methods can be used to configure access rights between individual ADS devices:

- Configuration of access rights via Access Control Lists [▶ 19] (using Mosquitto as an example)
- Configuration of access rights via a plugin [▶ 18] (only for Mosquitto)

4.5.1 Mosquitto plugin

A plugin was developed especially for the Mosquitto Message Broker to enable the definition of access rights between the individual TwinCAT ADS routers.
Please also note the system requirements [▶ 11] for operating this plugin.

The plugin is delivered with the TwinCAT installation under TwinCAT 3.1 Build 4024. Build 4026 requires the installation of the corresponding package (TwinCAT.XAE.MqttPlugin). The plugin is installed in the following directory and can be referenced from there in the Mosquitto configuration.

```
\TwinCAT\AdsApi\TcMqttPlugin
```

The plugin is available in a 32-bit and a 64-bit version, depending on which version you use of the Mosquitto Message Broker. The plugin is then integrated into the configuration of the Mosquitto Message Broker as follows:

```
auth_plugin <Path>TcMqttPlugin.dll
auth_opt_xml_file <Path>MyACL.xml
```

The *MyACL.xml* file contains the access configuration to the broker itself, as well as the configuration of the communication between the connected TwinCAT ADS routers. This configuration is explained in more detail in the following section.

Configuration

The plugin offers the option of configuring virtual AMS networks. To do this, specify which device can access which other device for each target device. In contrast to the classic ADS routes, these connections are directional: A target therefore does not also have the right to access the source.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd">
  <Ams>
    <Topic>VirtualAmsNetwork1</Topic>
    <User>
      <Name>EngineeringStation</Name>
    </User>
    <User>
      <Name>CX-12345</Name>
      <Access>EngineeringStation</Access>
    </User>
    <User>
      <Name>CX-56789</Name>
      <Access>EngineeringStation</Access>
    </User>
  </Ams>
</TcMqttAclConfig>
```

The name of the Ams network is defined within an <Ams> node. It is used in the MQTT topics employed for the identification of the networks. Individual <User> elements describe the devices. These elements have a <Name> attribute that describes the MQTT identity with which the connection was established. The identity can be transferred via various TLS mechanisms, e.g. via the TLS-PSK Identity or the CommonName (CN) of a client certificate. Our samples [▶ 21] here show possible configuration variants.

Access-authorized devices are defined via the <Access> element. In the sample above, the "EngineeringStation" identity has access to two CX devices, but the CX devices do not have access to the "EngineeringStation" or to each other.

The configuration file is reloaded cyclically so that a restart of the broker is not necessary.

Restrictions with regard to the AmsNetId to be registered

With this configuration each validly connected device can assume an arbitrary AmsNetId and thus an identity from the point of view of ADS. This can be further restricted as required:

```
<User>
  <Name>CX-56789</Name>
  <Access>EngineeringStation</Access>
  <NetId>192.168.56.1.1.1</NetId>
</User>
```

As soon as at least one NetId is specified, only one NetId can be registered from this list. An alternative solution would be to enter the NetId in the CommonName (CN) of the client certificate.

4.5.2 Mosquitto ACL

Many message brokers allow the configuration of Access Control Lists (ACLs) to restrict client interactions to certain topics. The following chapter shows this configuration using the Mosquitto Message Broker as an example. The procedure for granting access rights described here differs from that of the TcMqttPlugin [▶ 18]. This specifies which other ADS devices have access to an ADS device. With the (Mosquitto) ACL it is exactly the other way around, because here it is specified for an ADS device which other ADS devices it is allowed to access.

Overview

The Mosquitto Message Broker allows the configuration of an Access Control List, which is defined as a separate configuration file and referenced in the main configuration of the broker. This configuration entry is shown below as an example:

```
acl_file C:\Program Files\mosquitto\mosquitto.acl
```

You can also find a complete configuration file in our samples [▶ 21] for download.

In the ACL file, you can define authorizations for publishing and subscribing to certain topics and specify them separately for each user. The access rights for a user are always introduced by the following line:

```
user <username>
```

Subsequently the reading and writing rights are defined according to the following scheme:

```
topic [read|write|readwrite] <topicName>
```

Configuration for ADS-over-MQTT

For ADS-over-MQTT, two things must be ensured according to the communication flow [▶ 14]: access of all ADS devices to the discovery topics and sending/receiving via the communication topics.
The ADS device must always have read/write access to its own topic. The device receives read rights for the discovery topic of other ADS devices.
To exchange ADS messages, an ADS device must have read/write access to the communication topic of the devices. The ADS device is identified by its own identity on the message broker. This identity can, for example, originate from a PSK or correspond to the CommonName (CN) of a client certificate. The following configuration illustrates these relationships.

```
user <identity>
topic readwrite <VirtualAmsNetworkName>/<OwnAmsNetId>/#
topic read <VirtualAmsNetworkName>/+/info
topic readwrite <VirtualAmsNetworkName>/+/ams/#
```

If an ADS device is to be denied access to another device, it must be ensured that there are no write permissions for the topic with the target AmsNetId.

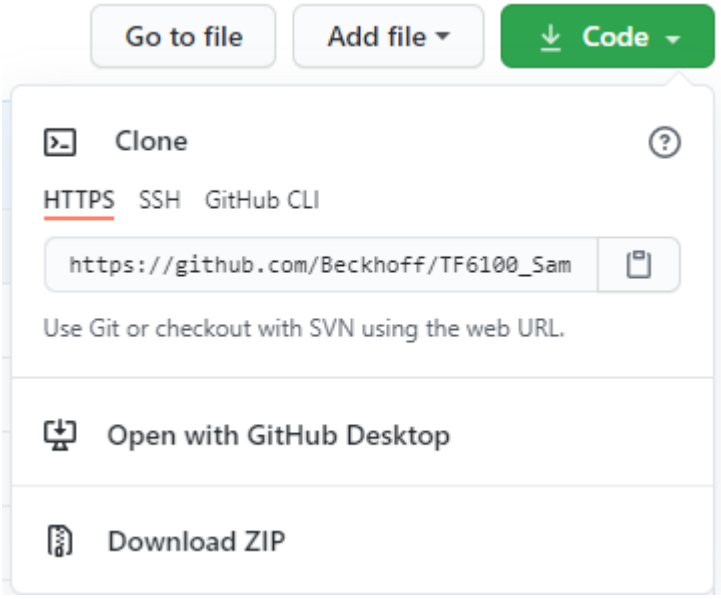
The familiar TcMqttPlugin option that an ADS device may register only one AmsNetId is also possible with the Mosquitto ACL. To do this, the entry <OwnAmsNetId> must be replaced by precisely one foreseen AmsNetId. If it is to be possible for the ADS device to register with an arbitrary AmsNetId, then the wildcard (#) has to be set for <OwnAmsNetId>.

The following is an example of the access rights entries for communication between two ADS devices:

```
user EngineeringStation
topic readwrite VirtualAmsNetwork1/18.153.78.19.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
user CX-12345
topic readwrite VirtualAmsNetwork1/3.120.15.8.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
```

5 Samples

Sample code and configurations for this product can be obtained from the corresponding repository on GitHub: https://github.com/Beckhoff/ADS-over-MQTT_Samples. There you have the possibility to clone the repository or download a ZIP file with the sample.



Trademark statements

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

Third-party trademark statements

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

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

More Information:
www.beckhoff.com/te1000

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