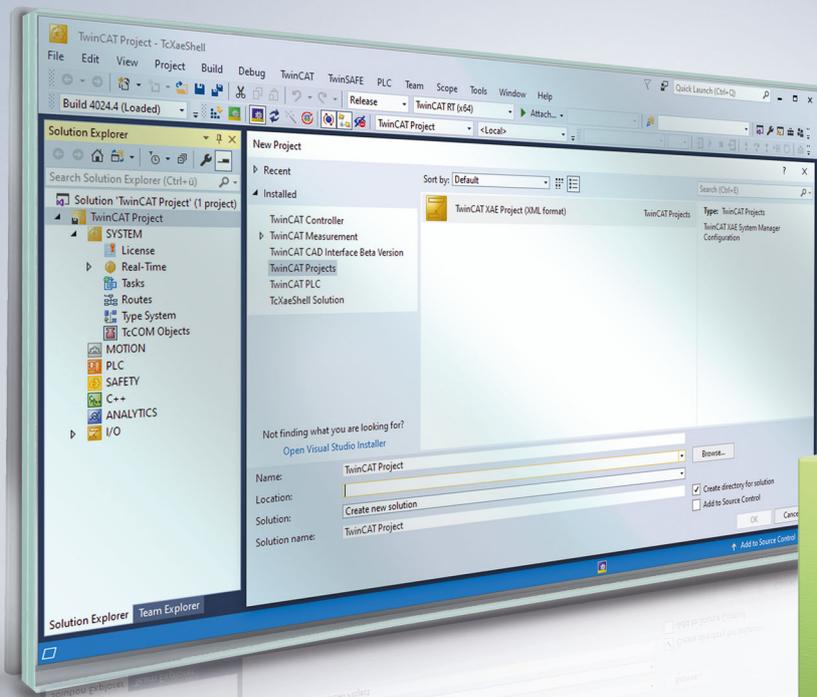


BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | ADS-over-MQTT



Inhaltsverzeichnis

1	Vorwort	5
1.1	Hinweise zur Dokumentation	5
1.2	Zu Ihrer Sicherheit	6
1.3	Hinweise zur Informationssicherheit	7
2	Allgemeine Beschreibung	8
3	Voraussetzungen	9
4	Technische Einführung	10
4.1	MQTT Grundlagen	10
4.2	Architektur	15
4.3	Transparente Nachrüstung	16
5	Konfiguration	18
5.1	TwinCAT	18
5.2	Broker	18
5.3	Security	20
5.3.1	TLS / PreSharedKey (PSK)	20
5.3.2	TLS / Zertifikate	22
6	Anwendungsszenarien	24
6.1	NAT-basierte Netzwerke	24
6.2	ADS Verschlüsselung	24
7	Beispiel	26
7.1	ADS-over-MQTT mit TLS und X.509 Zertifikaten	26
7.2	ADS-over-MQTT mit TLS und PSK	33

1 Vorwort

1.1 Hinweise zur Dokumentation

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

Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

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

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.

Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702

mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.



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

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.

Zuwendungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

GEFAHR

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

WARNUNG

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

VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt



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

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Allgemeine Beschreibung

„ADS over MQTT“ ist aus Sicht des ADS Protokolls ein neuer Transportkanal. Es werden also exakt die gleichen ADS Kommandos über MQTT übertragen, wie auch über andere Kommunikationsprotokolle.

Hierzu wird vom TwinCAT-Router eine Verbindung zum Broker aufgebaut, um Kommandos des ADS Protokolls sowohl abzusenden, wie auch zu empfangen.

Am lokalen Gerät wird damit also der Endpunkt des Brokers konfiguriert. Hieraus ergibt sich, dass sich die 1:1 Beziehung einer ADS-Route erst im Zusammenspiel mit der passenden Broker Konfiguration ergibt.



Diese Dokumentation gibt sowohl einen Überblick über die Möglichkeiten des Einsatzes, wie auch die technische Beschreibung wie ein „virtuelles ADS-Netzwerk“ über einen MQTT Message Broker konfiguriert werden kann.

Vorteile eines MQTT-basierten ADS-Netzwerks

- Subnetze, NAT-basierte Netzwerke und Firewalls:**
 In einem klassischen ADS-Aufbau werden eingehende TCP/IP-Verbindungen in beide Richtungen verwendet. Dieses bedingt, dass die Geräte im Normalfall im gleichen Netzwerk stehen müssen. In verteilten Anlagen mit unterschiedlichen Subnetzen ergeben sich aufwändige Konfigurationen, um die entsprechenden ADS-Routen nutzbar zu machen.
 Bei MQTT-basierten ADS-Netzwerken wird nur eine ausgehende TCP/IP-Verbindung von den Geräten verwendet. Hierdurch kann der Broker im überlagerten Netz zwischen allen Teilnehmern vermitteln. Durch die ausgehenden Verbindungen kann eine typische Firewall verwendet werden und es müssen keine eingehenden Ports hinterlegt werden.
- Zugriffskontrolle:**
 In einem klassischen ADS-Aufbau kann, nach Anlegen der entsprechenden Routen, eine bidirektionale Kommunikation durchgeführt werden.
 Ein Zugriff von Teilnehmer A der auf B zugreift, erlaubt also auch, dass Teilnehmer B auf A zugreifen kann.
 Im MQTT-basierten ADS-Netzwerk kann konfiguriert werden, dass ein Teilnehmer A auf B zugreifen kann, jedoch nicht andersrum.
- Security / Verschlüsselung:**
 Die Kommunikation von TwinCAT zu dem Broker kann durch TLS (mit Zertifikaten oder PreSharedKey (PSK)) verschlüsselt werden.

Nachteilig ist der erhöhte administrative Aufwand anzusehen. Dieser wird jedoch pro Teilnehmer bei einem größeren Netzwerk vertretbar gering ausfallen.

HINWEIS

ADS Zugriff bedeutet Vollzugriff

Wie im [Security Advisory 2017-01](#) beschrieben, bietet ADS Vollzugriff auf ein Gerät. Secure ADS bietet eine Autorisierung sowie Verschlüsselung für die Kommunikation, es stellt also eine Transport-Verschlüsselung dar. Wenn eine ADS Route besteht, existiert also Vollzugriff. Dezierten, Rollen-bezogenen Zugriff auf einzelne Daten bieten Lösungen, wie z. B. OPC-UA.

3 Voraussetzungen

● **TwinCAT 3.1 Build 4022.0 erforderlich**

i Das ADS-over-MQTT ist eine Erweiterung von Build 4022 und damit erst ab diesem Release verfügbar.

- ADS-over-MQTT ist Bestandteil von TC1000 und kann ohne Lizenzkosten genutzt werden.
- Die verwendeten Geräte benötigen eine ausgehende Netzwerkkommunikation zum Broker.
- Es muss ein MQTT Broker bereitgestellt werden, über den die Kommunikation erfolgen kann.
- Die bereitgestellte Erweiterung ist für den Eclipse Mosquitto Broker verfügbar.
- Zur TLS-Verschlüsselung müssen ggf. entsprechende Zertifikate generiert und signiert werden.

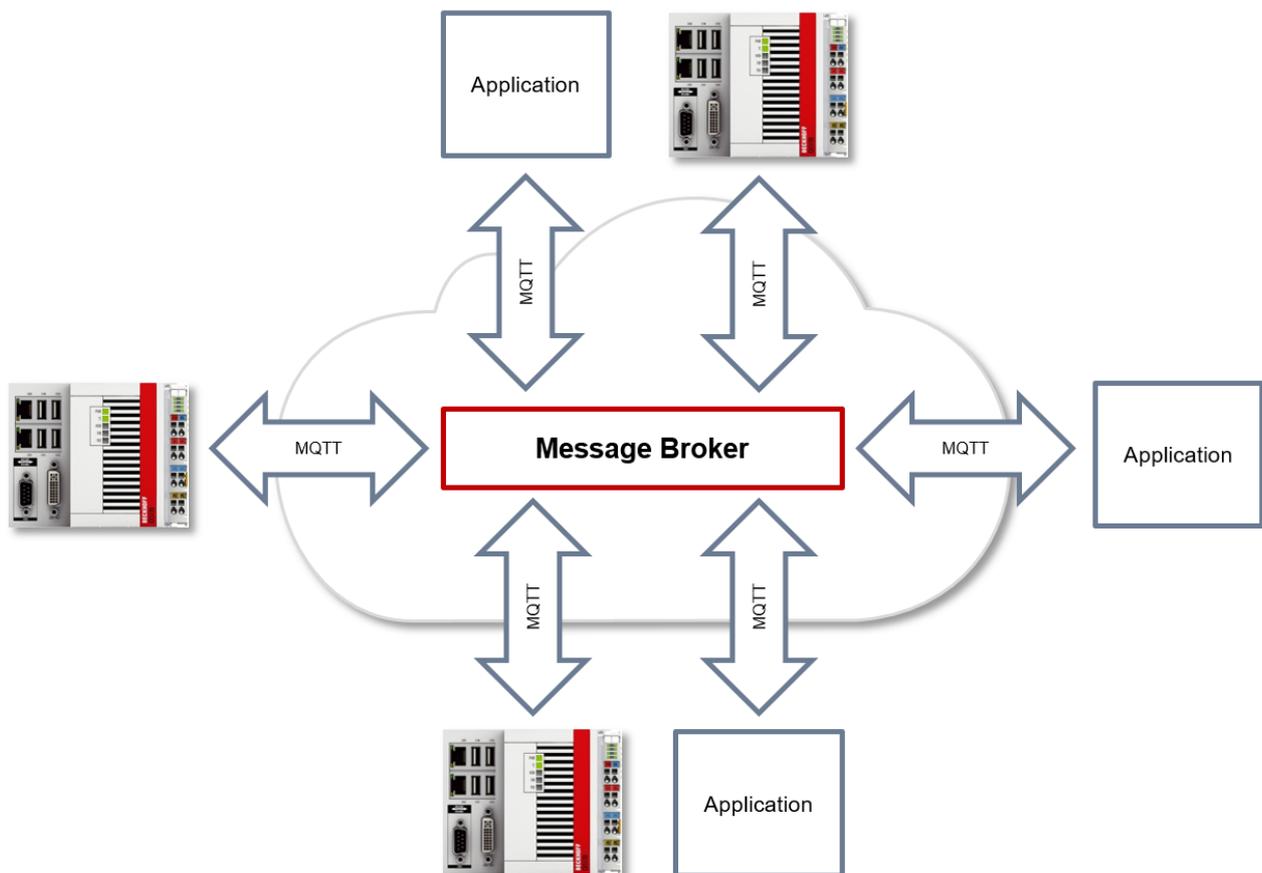
4 Technische Einführung

In diesem Abschnitt wird ein Überblick über die verwendeten Technologien sowie die grundsätzliche Architektur eines „virtuellen ADS-Netzwerks“ gegeben.

ADS-over-MQTT führt dafür einen zusätzlichen Kommunikationskanal ein, wodurch sich ADS-Routen über MQTT ergeben. Diesen können die Programme, die als ADS Devices auf den Geräten gestartet werden, nutzen, ohne dass sie modifiziert werden.

4.1 MQTT Grundlagen

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



ClientID

Beim Herstellen einer Verbindung mit dem Message Broker übermittelt der Client eine sogenannte ClientID, welche zur eindeutigen Identifizierung des Clients auf dem Message Broker dient. Der MQTT-Kommunikationstreiber aus TwinCAT3 erzeugt automatisch eine eigene ClientID, welche sich an dem folgenden Namensschema orientiert:

PlcProjectName-TcMqttClient%n

%n ist hierbei ein inkrementeller Zähler für die Nummer der jeweiligen MQTT Client Instanz. Jede Instanz des Funktionsbausteins FB_lotMqttClient erhöht hierbei diesen Zähler. In den meisten Fällen ist das Verwenden dieses ClientID Formats ausreichend. In speziellen Fällen, z.B. abhängig vom Message Broker

oder auch durch die eigene MQTT Applikation bedingt, muss eine anwendungsspezifische ClientID vergeben werden. Dies kann über einen entsprechenden Eingang an den Funktionsbausteinen FB_lotMqttClient und FB_lotMqtt5Client erfolgen.

Soll eine eindeutige ClientID automatisch beim Start des SPS Projekts generiert werden, so bietet sich die Verwendung einer GUID an, welche über den Funktionsbaustein FB_CreateGuid aus der Bibliothek Tc2_System erzeugt werden kann. Der folgende Beispielcode verdeutlicht die Verwendung dieses Funktionsbausteins.

```
PROGRAM MAIN
VAR
  fbGuid : FB_CreateGUID;
  objGuid : GUID;
  sGuid : STRING;
  nState : UINT;
  bStart : BOOL; // set to TRUE to start this sample
END_VAR

CASE nState OF
  0 :
    IF bStart THEN
      bStart := FALSE;
      nState := nState + 1;
    END_IF

  1 : // create GUID using FB_CreateGuid from Tc2_System library
    fbGuid(bExecute := TRUE, pGuidBuffer := ADR(objGuid), nGuidBufferSize := SIZEOF(objGuid));
    IF NOT fbGuid.bBusy THEN
      fbGuid(bExecute := FALSE);
      IF NOT fbGuid.bError THEN
        nState := nState + 1;
      ELSE
        nState := 255; // go to error state
      END_IF
    END_IF

  2: // GUID has been created, now convert to STRING
    sGuid := GUID_TO_STRING(objGuid);
    nState := nState + 1;

  3: // done

  255: // error state

END_CASE
```

Nach Ausführung dieser State Machine enthält die Variable sGuid die generierte GUID als STRING. Diese kann dann an den Funktionsbausteinen FB_lotMqttClient und FB_lotMqtt5Client als ClientID verwendet werden.

Payload

Der Nachrichtinhalt einer MQTT-Nachricht wird als sogenannter Payload bezeichnet. Es können beliebige Daten übertragen werden, beispielsweise ein Text, ein einzelner Zahlenwert oder eine gesamte Informationsstruktur.

● Message-Payload-Formatierung

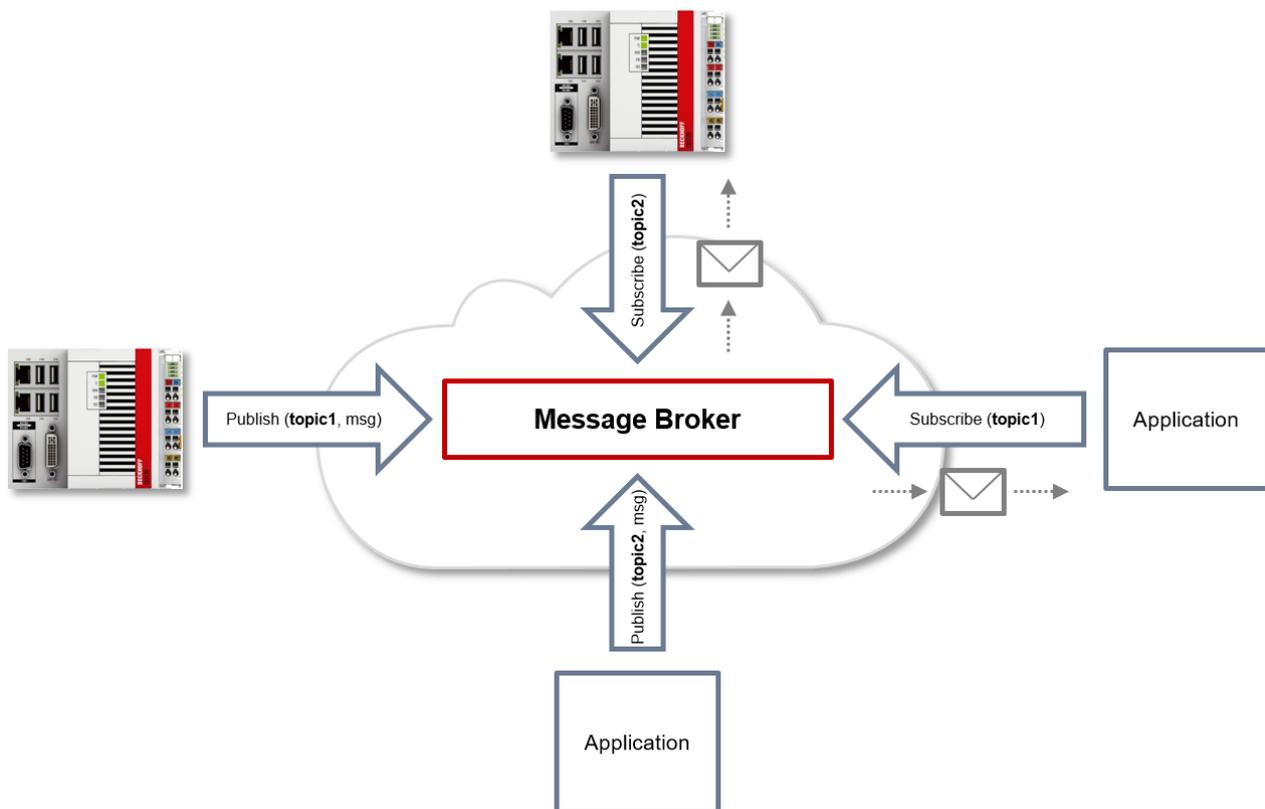
I Beachten Sie, dass der Datentyp und die Formatierung des Inhalts der Sender- und Empfängerseite bekannt sein müssen, insbesondere beim Versand von Binärinformationen (Alignment) oder Strings (mit/ohne Nullterminierung).

Topics

Bei Verwendung eines Message Brokers, welcher auf dem Protokoll MQTT basiert, wird das Senden (Publish) und Abonnieren (Subscribe) von Nachrichten mithilfe sogenannter Topics organisiert. Der Message Broker filtert eingehende Nachrichten anhand dieser Topics für jeden verbundenen Client. Ein Topic kann hierbei auch aus mehreren Ebenen bestehen, wobei die Ebenen durch ein „/“ voneinander getrennt sind.

Beispiel: Campus / Building1 / Floor2 / Room3 / Temperature

Der Publisher einer Nachricht gibt beim Versand immer an, für welches Topic eine Nachricht gedacht ist. Ein Subscriber hingegen gibt an, für welches Topic er sich interessiert. Der Message Broker leitet dann die Nachricht entsprechend weiter.



Beispielkommunikation 1 in der Grafik oben:

- Eine Applikation subscribed sich auf das Topic mit dem Namen „topic1“.
- Ein Controller published eine Nachricht an das Topic mit dem Namen „topic1“.
- Der Message Broker leitet die Nachricht entsprechend an die Applikation weiter.

Beispielkommunikation 2 in der Grafik oben:

- Ein Controller subscribed sich auf das Topic mit dem Namen „topic2“.
- Eine Applikation published eine Nachricht an das Topic mit dem Namen „topic2“.
- Der Message Broker leitet die Nachricht entsprechend an den Controller weiter.

Wildcards

Bei der Verwendung von Topics können auch sogenannte „Wildcards“ benutzt werden. Eine Wildcard ersetzt einen Teil des Topics. Ein Subscriber erhält dann ggf. Nachrichten aus mehreren Topics. Es werden zwei Arten von Wildcards unterschieden:

- Single Level Wildcards
- Multi Level Wildcards

Beispiel „Single Level Wildcard“:

Das +-Symbol beschreibt eine Single Level Wildcard. Wird es z.B. vom Subscriber wie folgt verwendet, so werden entsprechende Nachrichten an die Topics entweder vom Subscriber empfangen oder nicht empfangen.

- Der Empfänger subscribed sich auf Campus/Building1/Floor2+/Temperature
- Der Publisher sendet an Campus/Building1/Floor2/Room1/Temperature - OK
- Der Publisher sendet an Campus/Building1/Floor2/Room2/Temperature - OK
- Der Publisher sendet an Campus/Building42/Floor1/Room1/Temperature - NOK

- Der Publisher sendet an Campus/Building1/Floor2/Room1/Fridge/Temperature - NOK

Beispiel „Multi Level Wildcard“:

Das #-Symbol beschreibt eine Multi Level Wildcard. Wird es z.B. vom Subscriber wie folgt verwendet, so werden entsprechende Nachrichten an die Topics entweder vom Subscriber empfangen oder nicht empfangen. Das #-Symbol muss hierbei immer als letztes Symbol im Topic-String verwendet werden.

- Der Empfänger subscribed sich auf Campus/Building1/Floor2/#
- Der Publisher sendet an Campus/Building1/Floor2/Room1/Temperature - OK
- Der Publisher sendet an Campus/Building1/Floor2/Room2/Temperature - OK
- Der Publisher sendet an Campus/Building42/Floor1/Room1/Temperature - NOK
- Der Publisher sendet an Campus/Building1/Floor2/Room1/Fridge/Temperature – OK
- Der Publisher sendet an Campus/Building1/Floor2/Room1/Humidity - OK

QoS (Quality of Service)

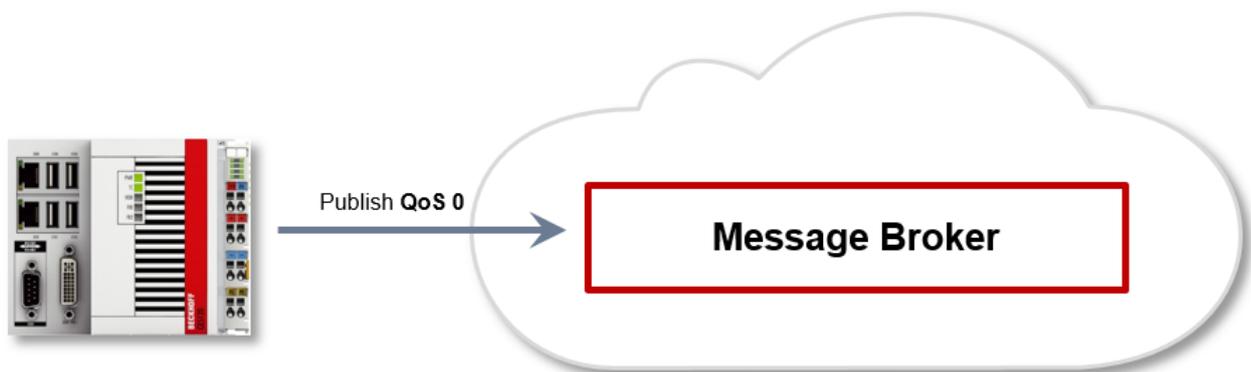
QoS ist eine Vereinbarung zwischen dem Sender und Empfänger einer Nachricht in Bezug auf das Garantieren der Nachrichtenübermittlung. Es existieren drei verschiedene Level in MQTT:

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

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

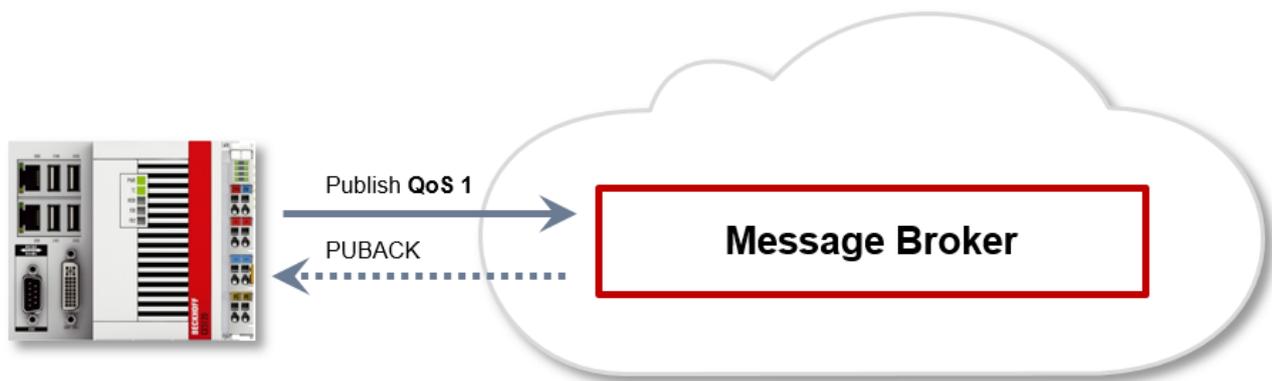
QoS-Level 0

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



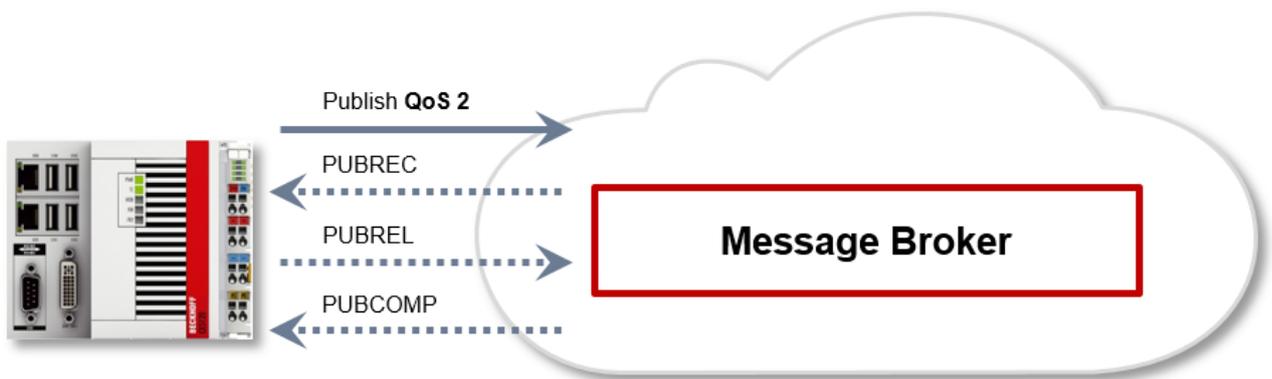
QoS-Level 1

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



QoS-Level 2

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



Der LastWill ist eine Nachricht, die im Falle eines irregulären Verbindungsabbruches vom Broker an alle Clients gesendet wird, die das passende Topic abonniert haben. Verliert der MQTT-Client in der SPS die Verbindung zum Broker und es wurde beim Verbindungsaufbau ein LastWill hinterlegt, so wird dieser LastWill vom Broker kommuniziert, ohne dass der Client sich darum kümmern muss.

Bei einem geplanten Disconnect wird der LastWill laut Spezifikation nicht zwingend übertragen. Aus Sicht des SPS-Programmierers kann dieser entscheiden, ob er vor Aufruf des Disconnects den LastWill publishen will. Dazu wird auf dem LastWill-Topic die LastWill-Nachricht noch einmal gepublished. Das ist notwendig, da der Broker aufgrund der regulären Verbindungsabbruches die Last Will-Nachricht nicht veröffentlichen würde.

Bei einem TwinCAT-Kontextwechsel und einem daraus folgenden Neustart der MQTT-Kommunikation sendet der IoT-Treiber den vorher spezifizierten LastWill an den Broker, weil in diesem Moment aus der SPS keine Möglichkeit mehr dazu besteht. Wenn bei Verbindungsherstellung kein LastWill definiert wurde, wird auch keine Nachricht vor dem Disconnect übertragen.

Sicherheit

Bei der Herstellung einer Verbindung zum Message Broker können hierbei auch Sicherheitsmechanismen, wie TLS eingesetzt werden, um die Kommunikationsverbindung zu verschlüsseln oder eine Authentifizierung zwischen Client und Message Broker zu realisieren.

Quellen

Für weitere und detailliertere Informationen zu MQTT empfehlen wir die folgenden Webseiten:

HiveMq Blog: <http://www.hivemq.com/blog/mqtt-essentials/> (Hauptgrundlage dieses Artikels)

4.2 Architektur

Der ADS Router in jedem Gerät übernimmt die Übermittlung der ADS-Kommandos zwischen den lokalen und auch entfernten „ADS Devices“.

Dieser Router kann so konfiguriert werden, dass ADS-Kommunikation auch über einen Broker erfolgen kann.

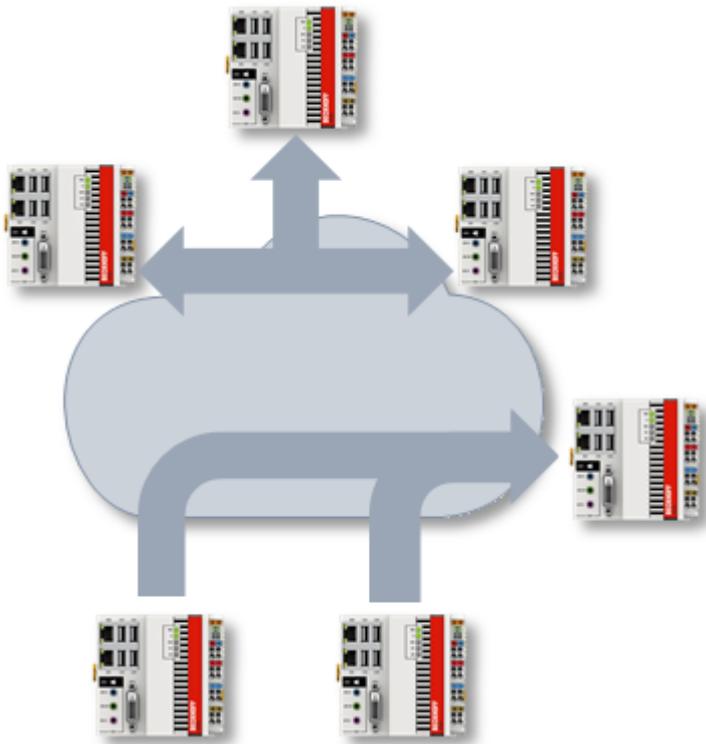
Der Broker übernimmt dabei eine Vermittlung der eintreffenden ADS-Kommandos anhand der hinterlegten Konfiguration.

Virtuelles AMS Netzwerk

Im Broker können unterschiedliche „Virtuelle AMS-Netzwerke“ mit unterschiedlichen Teilnehmern definiert werden. Jeder TwinCAT Router öffnet hierzu eine in seiner Konfiguration eingestellte MQTT- Verbindung zum Broker.

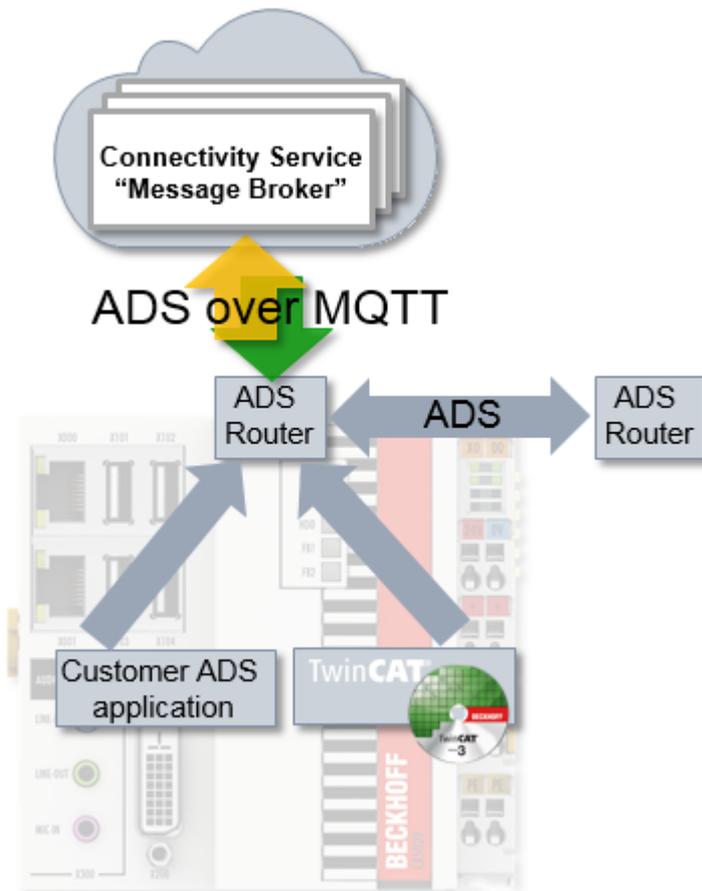
Im Broker wird konfiguriert, welche Geräte auf welche anderen Geräte zugreifen dürfen.

Insgesamt können virtuelle AMS-Netzwerke über einen Broker abgebildet werden.



Lokale Realisierung

Die Realisierung der ADS-over-MQTT-Anbindung erfolgt über den TwinCAT Router als zusätzlichen Transportkanal. Hierdurch ist die Erweiterung bezüglich der ADS-Clients wie auch die ADS-Server auf den jeweiligen Geräten transparent.



Technische Realisierung

Auf MQTT Protokoll-Ebene wird jeder ADS-Router als „User“ abgebildet, wobei dieses keine exklusive Beziehung darstellen muss.

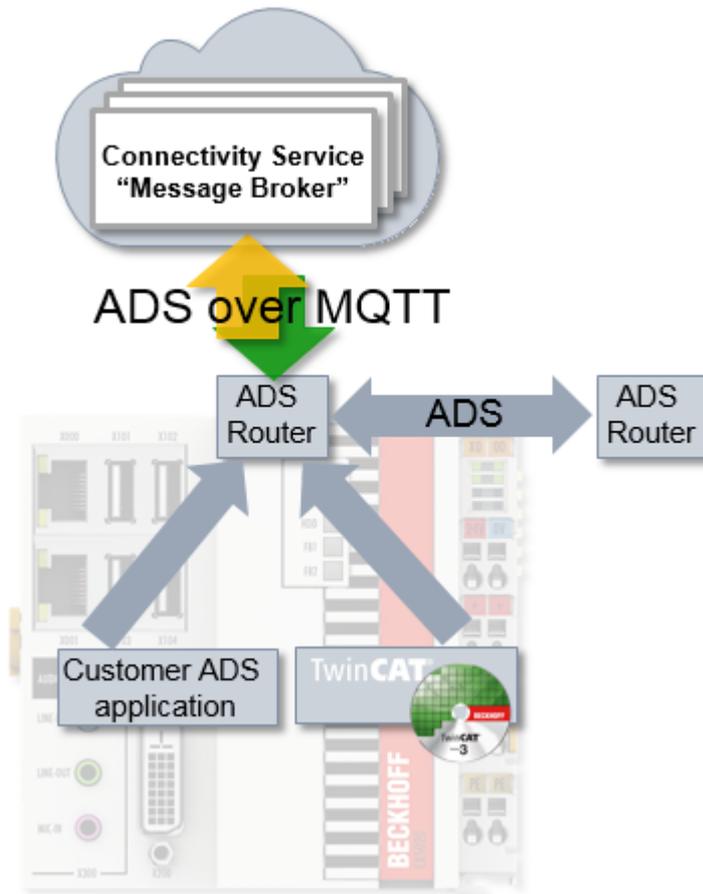
Es werden zwei unterschiedliche Topics-Kategorien von einem jeweiligen Kommunikationsteilnehmer verwendet:

- Discovery: **<NetworkName>/<AmsNetId>/info**
Ein sich verbindender Router sendet eine **RETAIN** Nachricht an dieses Topic, gleichzeitig subscribed er sich auf **<NetworkName>/+/info (QoS2)**, sodass er über weitere verbundene Router informiert wird.
- Kommunikation: **<NetworkName>/<AmsNetId>/ams/#**
Ein Router subscribed sich auf die **<NetworkName>/<AmsNetId>/ams/# (QoS2)**.
Die ADS-Kommandos werden zu diesem Router an **<NetworkName>/<AmsNetId>/ams** gesendet und die Antworten über **<NetworkName>/<AmsNetId>/ams/res**.

Somit ergibt sich, dass der Broker **RETAIN**-Topics sowie **QoS** implementieren muss, wie in der Einleitung beschrieben. Beispielfhaft sei hier der Eclipse Mosquitto-Broker genannt.

4.3 Transparente Nachrüstung

Durch die Realisierung von ADS-over-MQTT innerhalb des TwinCAT Routers wird ein nachträgliches „Retrofitting“ von Anwendungen ermöglicht. Alle ADS Anwendungen (Client und Server), dazu zählen auch Anwendungen die von Kunden geschrieben wurden, müssen nicht neu übersetzt werden.



Die ADS Anwendungen nutzen ADS Routen, um den Kommunikationspartner zu identifizieren. Diese ADS Route ist unabhängig vom Transportkanal und wird im TwinCAT Router beschrieben.

Wenn die genutzte Route auf eine ADS-over-MQTT Verbindung umgestellt wird, wird der ADS-Verkehr über den Broker (und damit ggf. abgesichert) transportiert.

5 Konfiguration

Die Konfiguration erfolgt über XML-Dateien sowohl auf Seiten des TwinCAT Systems, wie auch für den MQTT Broker.

5.1 TwinCAT

Der TwinCAT Router wird durch eine XML konfiguriert, um eine Verbindung mit einem oder mehreren Routern aufzunehmen.

Hierfür können die hier beschriebenen XML-Dateien mit beliebigen Namen unter dem Ordner C:\TwinCAT\3.x\Target\Routes (Windows CE: \Hard Disk\TwinCAT\3.x\Target\Routes) (x = TwinCAT Versionsnummer) abgelegt werden. Gespeicherte Änderungen werden übernommen, wenn der TwinCAT Router initialisiert wird, was beispielsweise bei den Übergängen von RUN->CONFIG oder auch CONFIG->CONFIG erfolgt.

Die XML Datei ist wie folgt aufgebaut:

```
<?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">BROKER-ADDRESS</Address>
      <Topic>VirtualAmsNetwork1</Topic>
      <User>CX-123456</User>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

Hierbei wird eine Verbindung aufgebaut und der TwinCAT-Router meldet sich mit dem gegebenen Namen (hier: CX-123456) bei dem Broker, welcher unter BROKER-ADDRESS erreichbar ist, und dem Port 1883 an. Die BROKER-ADDRESS ist dabei IP oder Name des Rechners auf dem der Broker läuft.

Dabei ist er Teilnehmer am Netzwerk „VirtualAmsNetwork1“ im Broker, welches sich in den verwendeten Topics wiederfindet, wie es unter [Architektur \[▶ 16\]](#) beschrieben ist.

Das Element <User> gibt dabei den User auf MQTT-Ebene an und kann im Broker z.B. in der [Mosquitto-Erweiterung \[▶ 18\]](#) genutzt werden, um Zugriffe zu konfigurieren.

Optional kann das <Mqtt>-Element ein Attribut ClientId tragen, um die MQTT-ClientID anzugeben. Diese wird ansonsten aus dem <User> und einem beliebigen String gebildet.

Diese Konfiguration baut eine unverschlüsselte Verbindung auf; Möglichkeiten der Verschlüsselung sind unter [Security \[▶ 20\]](#) dokumentiert.

5.2 Broker

Der MQTT Broker wird genutzt, um die ADS-Kommandos zwischen den Routern zu vermitteln. Die verwendete Topic-Struktur ist unter [Architektur \[▶ 16\]](#) beschrieben.

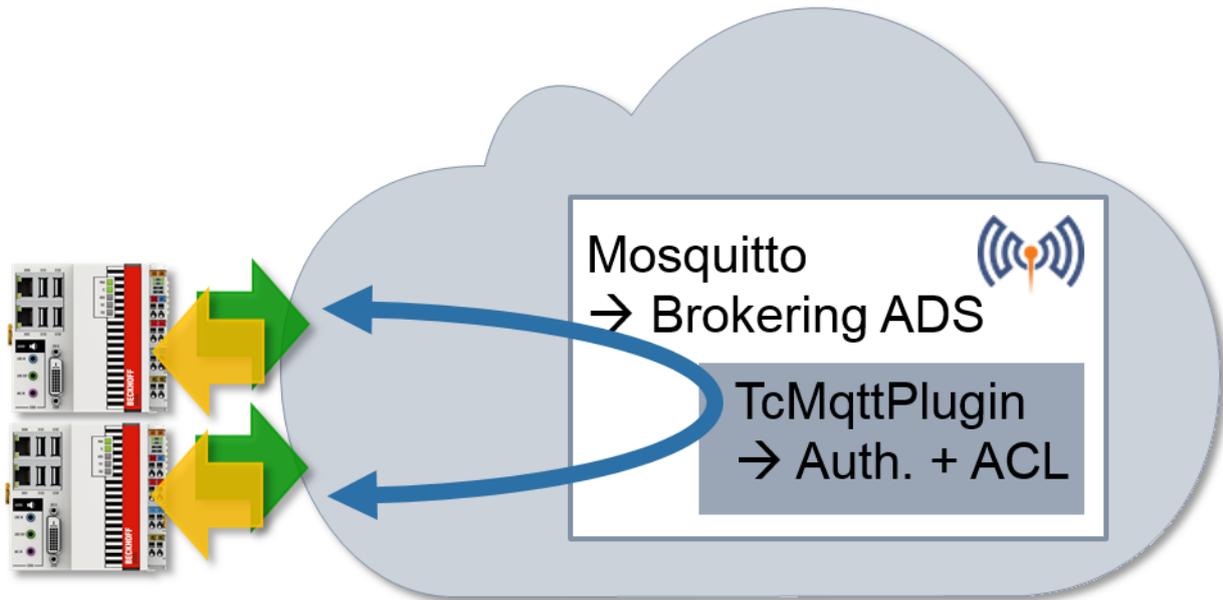
Allgemeines

Für ADS-over-MQTT kann ein beliebiger MQTT-Broker mit passender Unterstützung von beispielsweise RETAIN und QoS genutzt werden.

Ist dieser Broker im Sinne von Security schützenswert, da die ADS-Nachrichten schützenswert sind, müssen entsprechende Maßnahmen ergriffen werden. Die Security-Konfiguration auf Seiten von TwinCAT und beispielhaft auch für den Eclipse Mosquitto-Broker ist unter [Security \[▶ 20\]](#) beschrieben.

Tc-Plugin TcMqttPlugin.dll für den Eclipse Mosquitto Broker

Um in dem MQTT-Broker ein virtuelles Netzwerk von ADS-Geräten zu definieren, existiert eine Erweiterung für den [Eclipse Mosquitto Broker](#). Mittels dieser Erweiterung können sowohl Zugriffsrechte per PreSharedKey auf den Broker, wie auch Zugriffe zwischen den TwinCAT Routern mittels einer ACL (AccessControlList) eingestellt werden.



Das Plugin wird mit der TwinCAT-Installation ausgeliefert und befindet sich im Ordner `C:\TwinCAT\AdsApi\TcMqttPlugin` bzw. `C:\TwinCAT\AdsApi\x64\TcMqttPlugin`, falls ein 64-bit Mosquitto Broker eingesetzt wird.

In der Mosquitto-Konfiguration wird das Plugin wie folgt eingebunden:

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

Die Mosquitto-Konfigurationsdatei wird beim Starten des Mosquitto Brokers mittels des Parameters „-c“ angegeben, womit das Plugin inkl. der Konfiguration dann geladen wird.

Die Datei ACL.xml wird dabei in den folgenden Abschnitten beschrieben und stellt auf der einen Seite die Zugriffskonfiguration per PreSharedKey auf den Broker selbst bereit, wie aber auch die Konfiguration der Kommunikation zwischen den verbundenen TwinCAT Routern.

Konfiguration „Virtuelles AmsNetzwerk“

Das Plugin TcMqttPlugin bietet die Möglichkeit virtuelle AmsNetzwerke zu konfigurieren. Hierzu wird zu jedem Zielgerät angegeben, welches Gerät Zugriff auf welches andere Gerät hat.

Im Gegensatz zu den klassischen ADS-Routen sind diese Verbindungen gerichtet: Ein Ziel hat also nicht gleichzeitig das Recht auf die Quelle zuzugreifen.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
AnonymousLogin="true">
<!-- PSK Elements, if used -->
<Ams>
<Topic>VirtualAmsNetwork1</Topic>
<User>
<Name>EngineeringStation</Name>
</User>
<User>
<Name>CX-123456</Name>
<Access>EngineeringStation</Access>
</User>
<User>
<Name>CX-567890</Name>
<Access>EngineeringStation</Access>
</User>
</Ams>
</TcMqttAclConfig>
```

Innerhalb eines `<Ams>`-Knotens wird der Name des AmsNetzwerks definiert. Er wird in den genutzten MQTT Topics zur Identifizierung der Netzwerke verwendet.

Einzelne `<User>` Elemente beschreiben die Teilnehmer. Diese Elemente besitzen ein `<Name>`-Element, welches die MQTT-Identity beschreibt, mit der die Verbindung aufgebaut wurde – im Normalfall der Name

des Teilnehmers. Zusätzlich werden zugriffsberechtigte Geräte über das <Access>-Element definiert. In dem Beispiel hat „EngineeringStation“ also Zugriff auf zwei CX-Geräte, die CX-Geräte jedoch nicht auf die „EngineeringStation“ und auch nicht untereinander.

Die Datei wird zyklisch neu geladen, sodass ein Neustart des Brokers nicht nötig ist.

Da in dieser Erklärung keine Verschlüsselung vorgesehen ist, wird `AnonymousLogin="true"` verwendet.

Einschränkungen bzgl. der anzumeldenden AmsNetId

Bei dieser Konfiguration kann jedes gültig verbundene Gerät eine beliebige AmsNetID und damit Identität aus Sicht von ADS annehmen. Dieses kann eingeschränkt werden:

```
<User>
  <Name>CX-567890</Name>
  <Access>EngineeringStation</Access>
  <NetId>192.168.56.1.1.1</NetId>
</User>
```

Sobald mindestens eine NetId angegeben ist, kann nur eine NetId aus dieser Liste angemeldet werden.

Mosquitto Einstellungen

Im Zusammenhang mit der Konfiguration mittels TcMqttPlugin sind einige Einstellungen auf Seiten des Mosquitto Brokers wichtig zu beachten. Hierzu gehören:

- `psk_hint` Bezeichnet den `psk_hint` für den Verbindungsaufbau. Wird aktuell auf TwinCAT-Seite nicht überprüft.
- `port <1883|8883>` Der Port bezeichnet den Netzwerk-Port, den der Broker bereitstellt. Typischerweise ist 1883 unverschlüsselt und 8883 verschlüsselt.
- `require_certificate <true|false>` Bezeichnet die Notwendigkeit von Zertifikaten.
- `use_identity_as_username true` Bezeichnet, ob die Identity von Zertifikaten als Username auf Ebene von MQTT benutzt wird. Zur Nutzung des TcMqttPlugin wird dieses benutzt, sodass es als `true` anzulegen ist.

Minimal-Konfigurationsbeispiele sind in Abhängigkeiten von der genutzten TLS Verbindung in den entsprechenden Abschnitten beschrieben.

5.3 Security

Es stehen Möglichkeiten zur Absicherung der Kommunikation bereit. Hierfür kann eine TLS Verbindung auf Basis von X.509 Zertifikaten oder ein PreSharedKey (PSK) genutzt werden.

Insbesondere, wenn über nicht-vertrauenswürdige Netze (beispielsweise Internet) kommuniziert wird, wird empfohlen die Kommunikation mit TLS abzusichern. Der Broker selbst muss in einer vertrauenswürdigen Umgebung betrieben werden, da dort alle Nachrichten ungesichert sind.

i Kompromittierung des virtuellen ADS Netzwerks

Auch wenn die Kommunikation zwischen den Teilnehmern und dem Broker verschlüsselt über TLS erfolgt, existiert keine Absicherung der Teilnehmer untereinander. Die ADS-Kommandos liegen auf dem Broker unverschlüsselt vor.

Wenn ein Teilnehmer kompromittiert wurde, kann der Angreifer über die erlangten Rechte alle ADS-Kommandos ausführen. Zu diesen Kommandos gehören auch Datei-Lese-Operationen oder Operationen zum Starten von Prozessen.

5.3.1 TLS / PreSharedKey (PSK)

PreSharedKeys (PSK) sind Passwörter, die auf beiden Seiten einer Verbindung vorab durch einen Konfigurationsvorgang aufgebracht werden. Genutzt wird zur Kommunikation eine TLS 1.2 Verbindung.

TwinCAT Konfiguration mit PSK

Für einen TwinCAT Router kann in der Konfigurationsdatei für die Route mit einem PSK versehen werden, wobei der Key als HexString angegeben wird.

```
<?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">BROKER-ADDRESS</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Psk>
      <Identity>EngineeringStation</Identity>
      <Key>4D65696E5061737377C3B67274 [...]</Key>
    </Psk>
  </Mqtt>
</RemoteConnections>
</TcConfig>
```



Sicheres PSK

Ein sinnvoller PreSharedKey wird aus einem HexString von 64 Zeichen gebildet.

Alternativ kann der Key auch von TwinCAT bestimmt werden, damit eine einfachere Eingabe vorgenommen werden kann. Hierfür wird ein Passwort als normaler String im <Pwd> Element angegeben. TwinCAT berechnet aus diesem und der Identity mittels Sha256('Identity'+ 'Pwd') den zu nutzenden PSK. Ist das Attribut „IdentityCaseSensitive“ auf „false“ (oder nicht) gesetzt, wird die Identity als UpperCase-String für die Key-Berechnung verwendet.

```
<?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">BROKER</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Psk>
      <Identity>EngineeringStation</Identity>
      <Pwd IdentityCaseSensitive="false">!ABCDEFGHIjklmn123545</Pwd>
    </Psk>
  </Mqtt>
</RemoteConnections>
</TcConfig>
```

Minimale Mosquito-Konfiguration

Als einfachste Mosquito-Konfiguration können für PSKs folgende Einträge genutzt werden:

```
port 8883
psk_hint AHint
use_identity_as_username true
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file ACL.xml
```

Broker-Konfiguration mit PSK

Das Plugin TcMqttPlugin bietet die Möglichkeit einen PSK im Broker zu verwenden, um auf einen Broker zuzugreifen. Die Konfiguration ist in der Konfigurationsdatei des Plugins abgelegt, wobei der PSK als HexString angegeben wird.

Das IdentityCaseSensitive bietet die Möglichkeit, die Identitäten unabhängig von ihrer Groß-/Klein-Schreibweise zu betrachten.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
IdentityCaseSensitive="false">
  <Psk>
    <Identity>EngineeringStation</Identity>
    <Key>4D65696E5061737377C3B67274</Key>
  </Psk>
<!-- <Ams> Elements following -->
</TcMqttAclConfig>
```

Alternativ kann der Key auch von dem TcMqttPlugin bestimmt werden, damit eine einfachere Eingabe vorgenommen werden kann. Hierfür wird ein Passwort als normaler String im <PwD> Element angegeben. TwinCAT berechnet aus diesem und der Identity mittels Sha256('Identity'+PwD) den zu nutzenden PSK. Ist das Attribut auf Ebene von <TcMqttAclConfig> „IdentityCaseSensitive“ auf „false“ (oder nicht) gesetzt, wird die Identity als UpperCase-String für die Key-Berechnung verwendet.

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd"
IdentityCaseSensitive="false">
  <Psk>
    <Identity>EngineeringStation</Identity>
    <PwD>!ABCDEFGHIJKLMN123545</PwD>
  </Psk>
<!-- <Ams> Elements following -->
</TcMqttAclConfig>
```

5.3.2 TLS / Zertifikate

Zertifikate nach X.509 Standard können verwendet werden, um die entsprechende MQTT-Verbindung zum Broker abzusichern.

TwinCAT Konfiguration mit Zertifikaten

Für einen TwinCAT-Router können in den MQTT-Routen die Pfade zu den X.509 Zertifikaten konfiguriert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<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">BROKER-ADDRESS</Address>
      <Topic>VirtualAmsNetwork1</Topic>
      <Tls>
        <Ca>C:\TwinCAT\3.1\Target\Certificates\CA.crt</Ca>
        <Cert>C:\TwinCAT\3.1\Target\Certificates\Device.crt</Cert>
        <Key>C:\TwinCAT\3.1\Target\Certificates\Device.key</Key>
      </Tls>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

Hier werden im Element <Tls> die entsprechenden Pfade zu den Dateien angegeben. <Ca> ist dabei das X.509 Zertifikat der CertificateAuthority, also die ausstellende Stelle, von welcher Zertifikate akzeptiert werden sollen.

Die Elemente <Cert> sowie <Key> enthalten Pfade zu dem öffentlichen bzw. privaten Schlüssel des zu verwendenden Zertifikates.

- Der Hostname des Brokers („BROKER-ADDRESS“) muss zu dem Common Name des verwendeten Zertifikats passen. Dieses wird von den Clients überprüft.
- Der Common Name des Client Zertifikats wird als Identity in MQTT (und im TcMqttPlugin.dll) verwendet.

Minimale Mosquitto-Konfiguration

Als einfachste Mosquitto-Konfiguration können für die Nutzung von Zertifikaten folgende Einträge genutzt werden:

```
port 8883
cafile cert/CA.crt
certfile cert/Broker.crt
keyfile cert/Broker.key
require_certificate true
use_identity_as_username true
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file ACLCerts.xml
```

Broker-Konfiguration mit Zertifikaten

Die Identity, die in den <Ams> Elementen verwendet wird um das AmsNetzwerk zu beschreiben, wird über den CN des Zertifikates festgelegt.

Durch die CertificateAuthority ist festgelegt, welche Zertifikate Zugriff erhalten.

Eine zusätzliche Konfiguration auf Brokerseite entfällt damit.

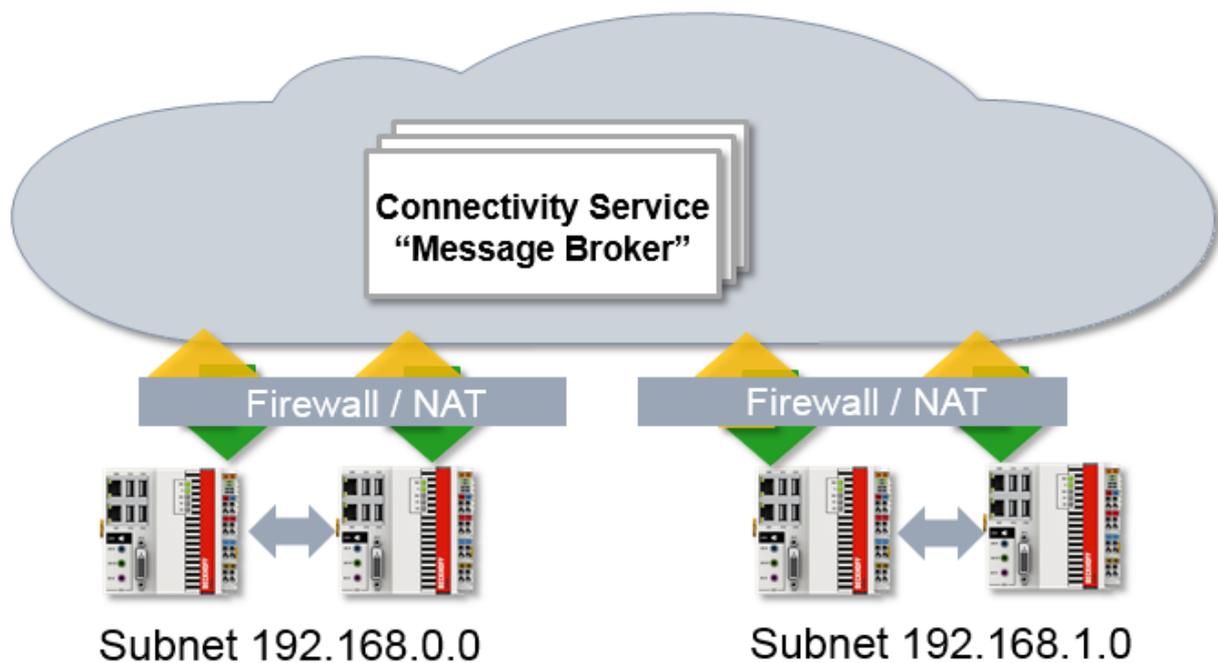
6 Anwendungsszenarien

An dieser Stelle sollen einige Anwendungsszenarien beschrieben werden, um den Mehrwert von ADS-over-MQTT darzustellen.

6.1 NAT-basierte Netzwerke

Durch die ausgehenden MQTT-Verbindungen von TwinCAT zu dem Broker wird eine einfache Kommunikation zwischen Subnetzen ermöglicht: Alle angeschlossenen Geräte müssen eine ausgehende Verbindung zu dem Broker aufbauen können – diese eine Verbindung wird für die gesamte ADS-Kommunikation verwendet. Der Broker ist die einzige Komponente mit eingehenden Verbindungen. Dies ist insbesondere bei Produktionsprozessen in großen, ggf. verteilten Anlagen von Vorteil. Hier werden häufig Subnetze mit NAT, Firewalls etc. eingesetzt. Trotzdem soll gelegentlich eine ADS-Kommunikation über die Netzwerkgrenzen hinweg ermöglicht werden.

In vielen Fällen ist eine solche Kommunikationsmöglichkeit wertvoll. Subnetze können aber auch aus Security-Gründen eingerichtet werden, sodass eine Kommunikation nicht gewünscht ist (Stichwort: „Zoning“).

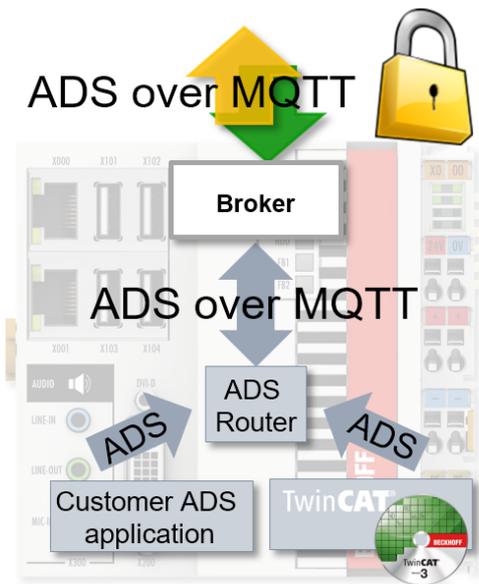


6.2 ADS Verschlüsselung

Durch die Fähigkeit von TLS, welches für eine Verschlüsselung von MQTT auf Transportebene genutzt wird, kann MQTT auch verwendet werden, um eine verschlüsselte ADS-Kommunikation zu ermöglichen. Dafür kann der Broker lokal auf den PC-basierten Steuerungen installiert werden. Der Broker wird dabei so konfiguriert, dass er lediglich die lokale Steuerung als Zugriffspunkt in einem virtuellen AmsNetzwerk anbietet.

Wird dann eine Verschlüsselung aktiviert und genutzt, entsteht eine auf TLS basierende, abgesicherte Verbindung per ADS zu einem TwinCAT-System.

Wenn diese genutzt wird, können ADS-Verbindungen durch die Firewall geblockt werden.

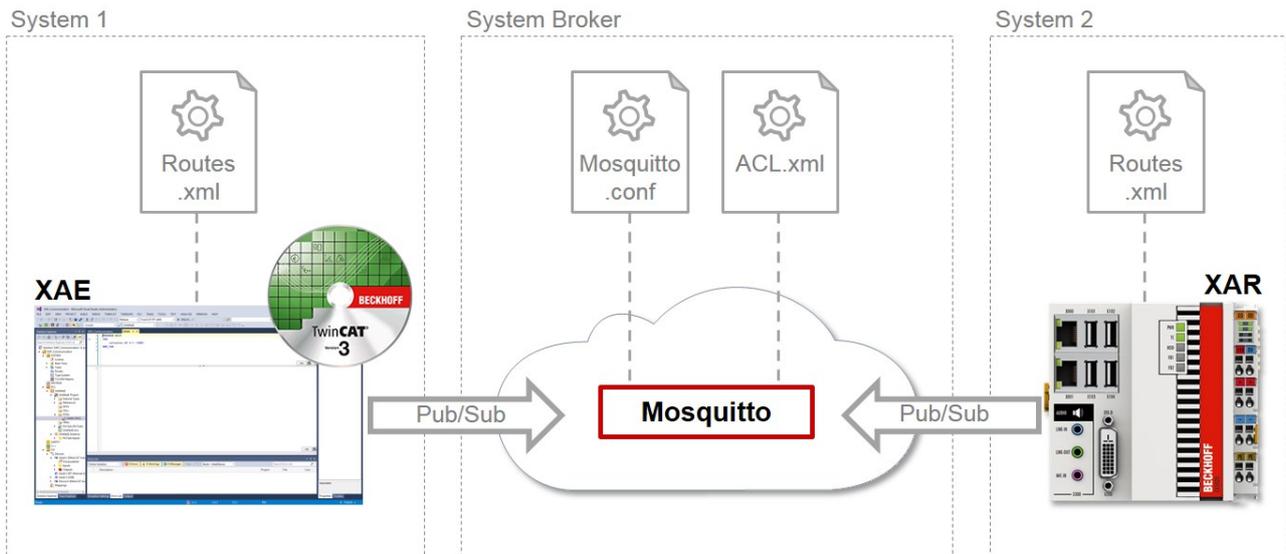


7 Beispiel



Dieses Beispiel stellt lediglich einen Workflow dar, um eine Testumgebung aufzusetzen. Sämtliche Parameter wie Zertifikatlaufzeiten, Schlüssellängen etc. sind entsprechend der realen Umgebung und Anwendung vorzunehmen.

Die Funktionsweise und Konfiguration von ADS-over-MQTT soll im Folgenden anhand eines Beispiels genauer erläutert werden. In dem Beispiel wird als Message Broker der Eclipse Mosquitto Broker verwendet sowie OpenSSL zur Erstellung der Zertifikate. Der Mosquitto Message Broker soll den Datenaustausch zwischen einem TwinCAT XAE und einer TwinCAT XAR realisieren. Um die Kommunikation abzusichern, wird das Verschlüsselungsprotokoll TLS in Kombination mit X.509 Zertifikaten oder TLS in Kombination mit PSK genutzt. Im nachfolgenden Bild ist der Aufbau des Anwendungsbeispiels schematisch dargestellt.



Zur Nutzung von ADS-over-MQTT sind die im Bild aufgeführten Konfigurationsdateien zu erstellen und entsprechend anzupassen. Im Folgenden werden zwei Beispiele vorgestellt.

In dem ersten wird ADS-over-MQTT mit TLS und X.509 Zertifikaten [► 26] konfiguriert und in dem zweiten mit TLS und PreSharedKeys (PSK) [► 33].

7.1 ADS-over-MQTT mit TLS und X.509 Zertifikaten

In diesem Abschnitt wird ein Beispiel zur Konfiguration von ADS-over-MQTT mit PSK- und X.509-Zertifikaten eingeführt. Die einzelnen Schritte zur Realisierung der Kommunikationsschnittstelle sind:

- ✓ TwinCAT 3.1 Build 4022.0 oder höher ist auf System 1 als XAE-Version installiert und auf System 2 als XAR-Version.
- 1. Erzeugen Sie die Zertifikate zur sicheren Kommunikation per TLS. Hierfür verwenden Sie das Programm OpenSSL, das Sie unter <https://www.openssl.org/source/> laden und anschließend installieren.

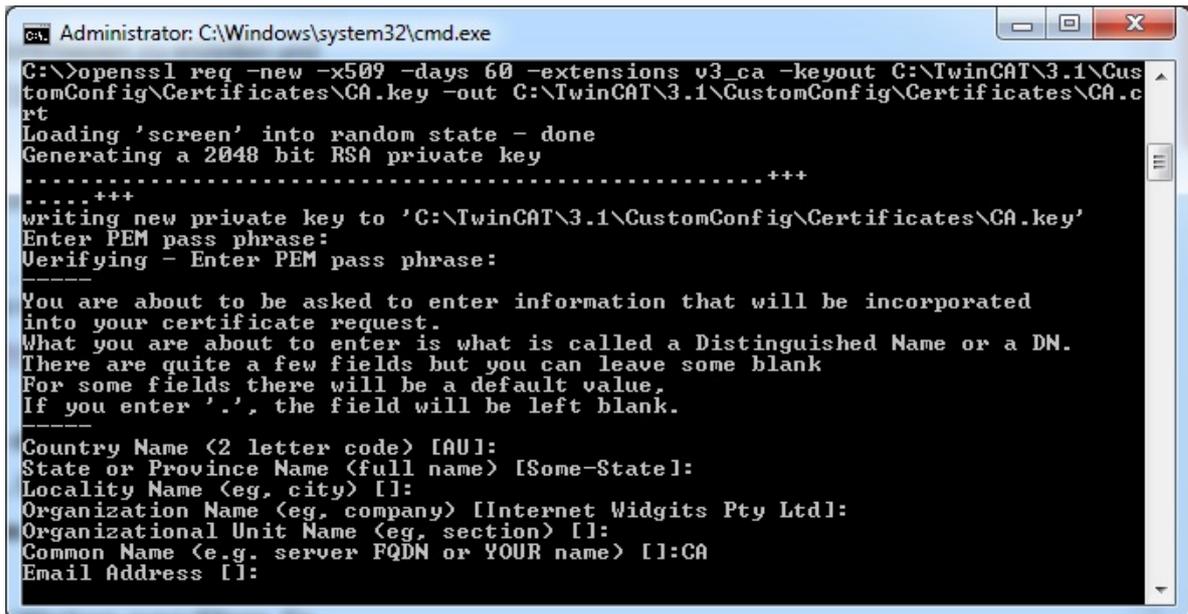
Hinweis Bei einem Windows-Betriebssystem muss der Pfad, in dem sich die Konfigurationsdatei von OpenSSL befindet, als Umgebungsvariable gesetzt sein. Führen Sie dies im Kommandozeilenprogramm eines x64-Systems mit folgendem Befehl durch: `set`

```
OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg
```

Mit Abschluss der Installation führen Sie das Kommandozeilenprogramm von Windows aus. Es wird mit der Generierung des CA-Zertifikats (Certificate Authority) begonnen. Dabei wird die Eingabe eine Passphrase verlangt. Geben Sie diese ein, merken Sie sich und geben Sie weitere Informationen zum CA an. Der entsprechende Befehl zur Generierung des CA Zertifikats lautet wie folgt:

```
openssl req -new -x509 -days 60 -extensions v3_ca -keyout C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -out C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt
```

⇒ Im Kommandozeilenprogramm sollte das Ergebnis wie folgt aussehen:



```

Administrator: C:\Windows\system32\cmd.exe
C:\>openssl req -new -x509 -days 60 -extensions v3_ca -keyout C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -out C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
++++
writing new private key to 'C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:CA
Email Address []:
  
```

- Erstellen Sie das Broker Zertifikat. Wichtig ist hierbei als CN (Common Name) den Hostnamen oder die IP-Adresse des Systems zu verwenden, auf dem der Mosquitto Message Broker betrieben werden soll. Zudem ist zu beachten, dass das System über die IP-Adresse bzw. den Hostnamen von den Clients erreichbar ist. Zur Generierung der Broker Zertifikate sind folgende Befehle im Kommandozeilenprogramm auszuführen:

Erstellen des Zertifikates:

```
openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key 2048
```

Erstellen des Certificate Signing Requests:

```
openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key -new
```

Signieren des CSR durch die zuvor erstellte CA, wofür das Passwort benötigt wird, welches bei der Erstellung der CA angegeben wurde:

```
openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -CA C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.crt -days 60
```

⇒ Im Kommandozeilenprogramm sollte das Ergebnis wie folgt aussehen:

```

Administrator: C:\Windows\system32\cmd.exe
C:\>openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key 2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)

C:\>openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key -new
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.8
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\>openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\broker.csr -CA
C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:\TwinCAT\3.1\CustomCo
nfig\Certificates\CA.key -CAcreateserial -out C:\TwinCAT\3.1\CustomConfig\Certif
icates\broker.crt -days 60
Loading 'screen' into random state - done
Signature ok
subject=C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=192.168.1.8
Getting CA Private Key
Enter pass phrase for C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key:

```

3. Generieren Sie die beiden Client Zertifikate für das TwinCAT XAE und TwinCAT XAR. Die OpenSSL-Befehle dazu sind im Folgenden angegeben.

Generierung des XAE Zertifikats:

```
openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key 2048
```

Erstellen des CSR:

```
openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key -new
```

Signieren des CSR durch die zuvor erstellte CA, wofür das Passwort benötigt wird, welches bei der Erstellung der CA angegeben wurde:

```
openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -CA C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.crt -days 60
```

Generierung des XAR Zertifikats:

```
openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key 2048
```

Erstellen des CSR:

```
openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key -new
```

Signieren des CSR durch die zuvor erstellte CA, wofür das Passwort benötigt wird, welches bei der Erstellung der CA angegeben wurde:

```
openssl x509 -req -in C:
```

```
\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -CA C:  
\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:  
\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:  
\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.crt -days 60
```

- ⇒ Im Kommandozeilenprogramm sollte das Ergebnis wie folgt aussehen:
TwinCAT XAE:

```

Administrator: C:\Windows\system32\cmd.exe
C:\>openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key
2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)

C:\>openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key -new
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:TwinCAT_XAE
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\>openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.csr -CA C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.crt -days 60
Loading 'screen' into random state - done
Signature ok
subject=/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=TwinCAT_XAE
Getting CA Private Key
Enter pass phrase for C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key:

```

TwinCAT XAR:

```

Administrator: C:\Windows\system32\cmd.exe
C:\>openssl genrsa -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key
2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)

C:\>openssl req -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -key C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.key -new
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:TwinCAT_XAR
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\>openssl x509 -req -in C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.csr -CA C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt -CAkey C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key -CAcreateserial -out C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAR.crt -days 60
Loading 'screen' into random state - done
Signature ok
subject=/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd/CN=TwinCAT_XAR
Getting CA Private Key
Enter pass phrase for C:\TwinCAT\3.1\CustomConfig\Certificates\CA.key:

```

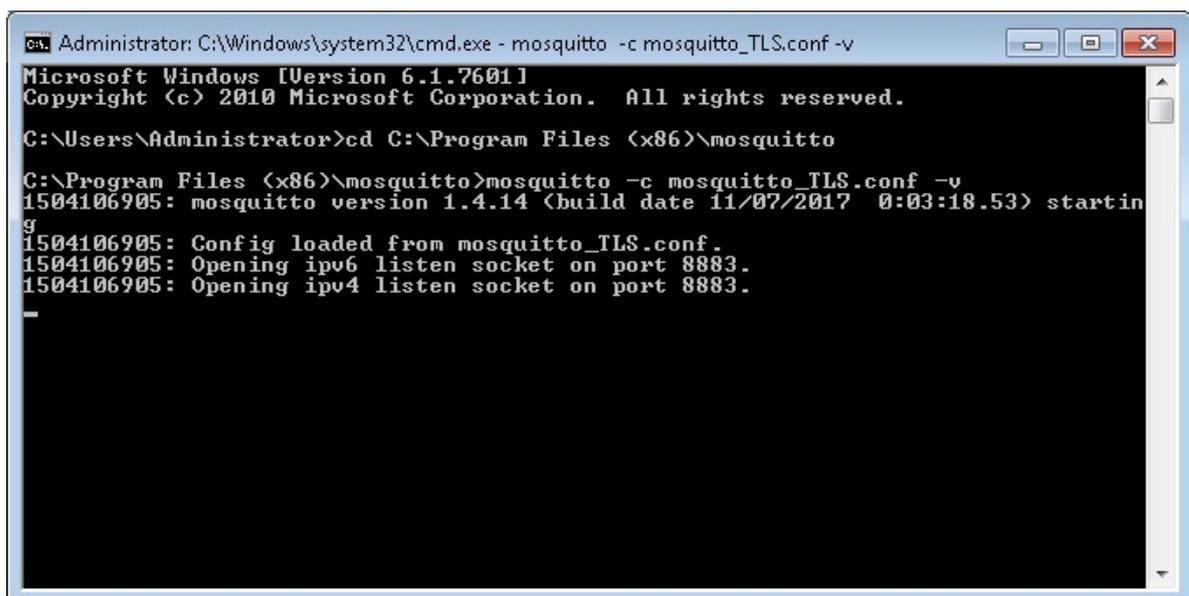
- Im Anschluss an die Generierung der Zertifikate installieren Sie den Mosquitto Broker. Laden Sie ihn von <https://mosquitto.org/download/> und installieren Sie ihn auf dem entsprechenden System.
- Nach der Installation des Mosquitto Brokers erstellen Sie für diesen die Konfigurationsdatei `mosquitto_TLS.conf` zur Verwendung von TLS mit Zertifikaten. Als Speicherort wählen Sie das Installationsverzeichnis von Mosquitto (Standardmäßig: `C:\Program Files (x86)\mosquitto`). Die Konfigurationsdatei soll folgende Einträge enthalten:

```
port 8883
allow_anonymous false
require_certificate true
use_identity_as_username true
cafile C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt
certfile C:\TwinCAT\3.1\CustomConfig\Certificates\broker.crt
keyfile C:\TwinCAT\3.1\CustomConfig\Certificates\broker.key
auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file C:\TwinCAT\AdsApi\TcMqttPlugin\ACL.xml
```

- Nun starten Sie den Mosquitto Message Broker über das Kommandozeilenprogramm von Windows. Dazu wechseln Sie in das Installationsverzeichnis des Mosquittos und führen Sie den nachfolgend aufgeführten Befehl aus. Bei dem Befehl sorgt `-v` für eine Ausgabe der Nachrichten, die durch den Broker versendet oder auch abgelehnt werden. Diese Option ist gerade während Tests sinnvoll zu nutzen.

```
mosquitto -c mosquitto_TLS.conf -v
```

⇒ Anschließend sollte das Ergebnis wie folgt aussehen:



```
Administrator: C:\Windows\system32\cmd.exe - mosquitto -c mosquitto_TLS.conf -v
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2010 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd C:\Program Files (x86)\mosquitto

C:\Program Files (x86)\mosquitto>mosquitto -c mosquitto_TLS.conf -v
1504106905: mosquitto version 1.4.14 (build date 11/07/2017 0:03:18.53) starting
1504106905: Config loaded from mosquitto_TLS.conf.
1504106905: Opening ipv6 listen socket on port 8883.
1504106905: Opening ipv4 listen socket on port 8883.
```

- Erstellen Sie für den Mosquitto als nächstes die `ACL.xml`, in der die Zugriffsrechte der Clients festgelegt werden. Legen Sie sie im Verzeichnis `C:\TwinCAT\AdsApi\TcMqttPlugin\` ab. Nehmen Sie in der `ACL.xml` die folgenden Einträge vor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>TwinCAT_XAE</Name>
  </User>
  <User>
    <Name>TwinCAT_XAR</Name>
    <Access>TwinCAT_XAE</Access>
```

```

    </User>
</Ams>
</TcMqttAclConfig>

```

8. Konfigurieren Sie nun das TwinCAT XAE und TwinCAT XAR für ADS-over-MQTT. Dazu erstellen Sie auf beiden Systemen im Verzeichnis C:\TwinCAT\3.x\Target\ einen Ordner „Routes“ und erzeugen Sie in diesem eine Datei mit dem Namen „MyRoute.xml“ (der Dateiname ist dabei beliebig). Folgend ist der Inhalt der Datei vom TwinCAT XAE dargestellt. Passen Sie für das TwinCAT XAR im <Cert>- und <Key>-Feld die Pfade entsprechend an. Wichtig ist immer, dass im <Address>-Feld der gleiche Eintrag angegeben wird, wie für den CN des Mosquitto Broker Zertifikats.

```

<?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">192.168.1.8</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Tls>
            <Ca>C:\TwinCAT\3.1\CustomConfig\Certificates\CA.crt</Ca>
            <Cert>C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.crt</
Cert>
            <Key>C:\TwinCAT\3.1\CustomConfig\Certificates\TwinCAT_XAE.key</
Key>
        </Tls>
    </Mqtt>
</RemoteConnections>
</TcConfig>

```

9. Damit die gespeicherte Konfiguration von ADS-over-MQTT für die TwinCAT-Systeme wirksam wird, initialisieren Sie jeweils den TwinCAT Router neu. Dies geschieht, indem Sie vom RUN-Mode in den CONFIG-Mode schalten oder vom CONFIG-Mode erneut in den CONFIG-Mode.

- ⇒ Zuletzt prüfen Sie, ob vom XAE eine Verbindung zum XAR aufgebaut werden kann. Ist dies der Fall, sollten die Ausgaben des Mosquitto Message Brokers wie folgt aussehen:

```

ca. Select Administrator: C:\Windows\system32\cmd.exe - mosquitto -p 8883 -v -c mosquitto_TLS.conf
172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (304 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (304 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (92 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (92 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetworkk1/172.17.214.125.1.1/ams/res', ... (36 bytes))

```

- ⇒ Damit ist ADS-over-MQTT mit TLS auf Basis von Zertifikaten für TwinCAT XAE und XAR erfolgreich eingerichtet.

7.2 ADS-over-MQTT mit TLS und PSK

Neben der Verwendung von TLS mit Zertifikaten kann MQTT-over-ADS auch auf Basis von PSK (Pre Shared Key) konfiguriert werden. Für diesen Anwendungsfall soll ebenfalls ein kurzes Beispiel eingeführt werden, dass bei der Umsetzung unterstützen soll. Es sind folgende Schritte durchzuführen:

1. Zunächst erstellen Sie die Konfigurationsdatei des Mosquittos (mosquitto_PSK.conf) in dessen Installationsverzeichnis (Standardmäßig: C:\Program Files (x86)\mosquitto). Danach nehmen Sie in der Datei die folgenden Einträge vor:

```

auth_plugin C:\TwinCAT\AdsApi\TcMqttPlugin\TcMqttPlugin.dll
auth_opt_xml_file C:\TwinCAT\AdsApi\TcMqttPlugin\ACL.xml
port 8883
psk_hint something
use_identity_as_username true

```

2. Im nächsten Schritt führen Sie den Mosquitto Message Broker aus. Der entsprechende Befehl dazu lautet:

```

mosquitto -c mosquitto_PSK.conf -v

```

3. In die ACL.xml tragen Sie die Schlüssel für das TwinCAT XAR und XAE ein:

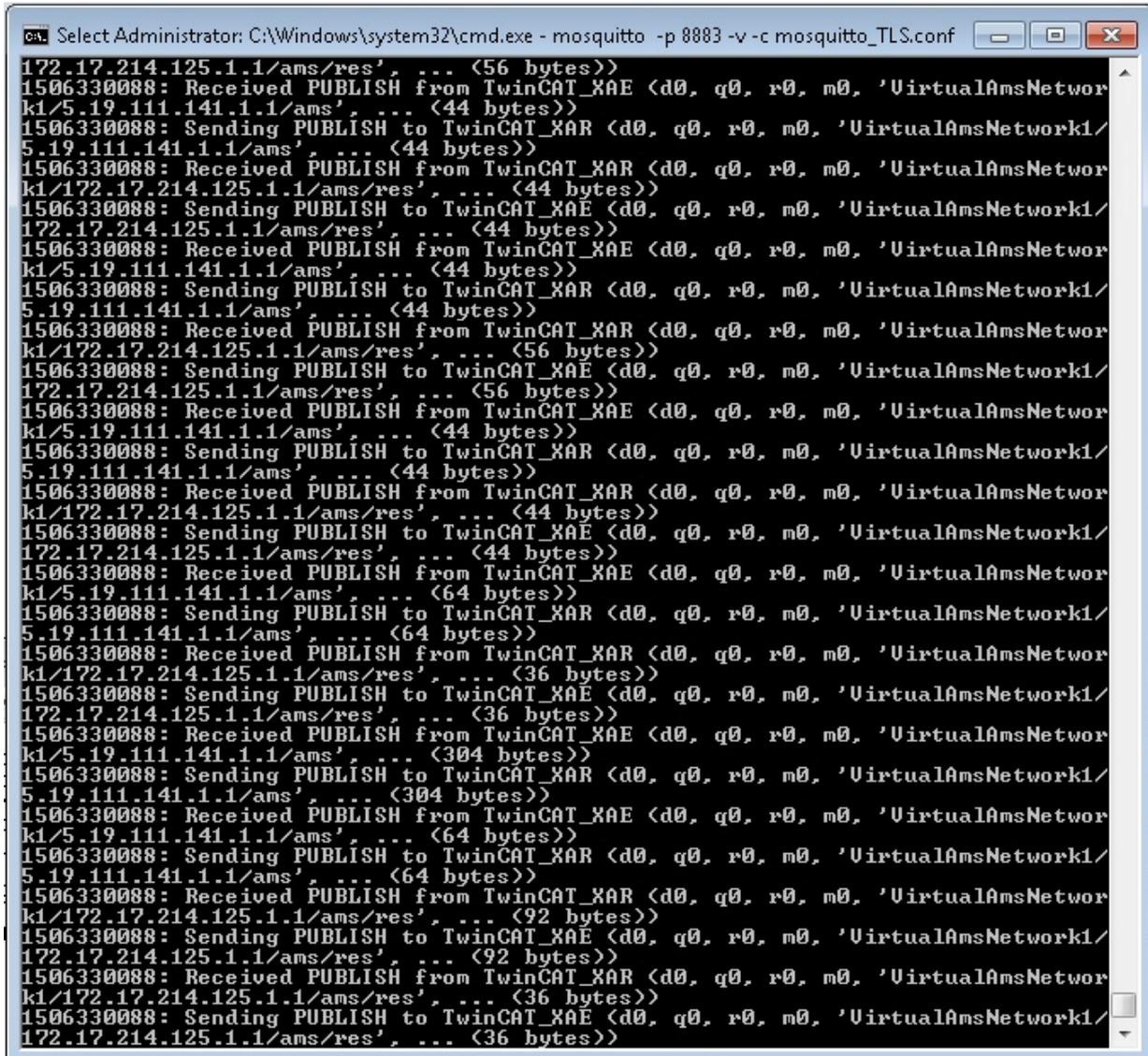
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:
\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd">
<Psk>
  <Identity>TwinCAT_XAE</Identity>
  <Pwd>abcdef1234!</Pwd>
</Psk>
<Psk>
  <Identity>TwinCAT_XAR</Identity>
  <Pwd>ghijkl5678?</Pwd>
</Psk>
<Ams>
  <Topic>VirtualAmsNetwork1</Topic>
  <User>
    <Name>TwinCAT_XAE</Name>
  </User>
  <User>
    <Name>TwinCAT_XAR</Name>
    <Access>TwinCAT_XAE</Access>
  </User>
</Ams>
</TcMqttAclConfig>
```

4. Machen Sie die in der ACL.xml definierten Schlüssel ebenfalls dem TwinCAT XAR und XAE bekannt. Dazu passen Sie auf beiden Systemen die Routes.xml im Verzeichnis C:\TwinCAT\3.x\Target\Routes an bzw. erstellen Sie sie. Für das TwinCAT XAE sind die Einträge nachfolgend aufgeführt:

```
<?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">192.168.1.8</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    <Psk>
      <Identity>TwinCAT_XAE</Identity>
      <Pwd>abcdef1234!</Pwd>
    </Psk>
  </Mqtt>
</RemoteConnections>
</TcConfig>
```

5. Die Einträge für das TwinCAT XAR sind fast identisch. Passen Sie lediglich die Werte der Felder <Identity> und <Pwd> entsprechend der Angaben in der ACL.xml an.
6. Nachdem auf beiden Systemen die Routes.xml fertig konfiguriert ist, initialisieren Sie jeweils den TwinCAT Router neu. Dazu schalten Sie vom RUN-Mode in den CONFIG-Mode oder vom CONFIG-Mode erneut in den CONFIG-Mode.

7. Abschließend prüfen Sie anhand der Ausgaben des Mosquitto Message Brokers, ob sich beide Systeme mit dem Broker verbinden können:



```
ca. Select Administrator: C:\Windows\system32\cmd.exe - mosquitto -p 8883 -v -c mosquitto_TLS.conf
172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (56 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (44 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (304 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (304 bytes))
1506330088: Received PUBLISH from TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork1/
5.19.111.141.1.1/ams', ... (64 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (92 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (92 bytes))
1506330088: Received PUBLISH from TwinCAT_XAR (d0, q0, r0, m0, 'VirtualAmsNetwork
k1/172.17.214.125.1.1/ams/res', ... (36 bytes))
1506330088: Sending PUBLISH to TwinCAT_XAE (d0, q0, r0, m0, 'VirtualAmsNetwork1/
172.17.214.125.1.1/ams/res', ... (36 bytes))
```

- ⇒ Damit ist ADS-over-MQTT mit TLS auf Basis von PSK für TwinCAT XAE und XAR erfolgreich eingerichtet.

Mehr Informationen:
www.beckhoff.com/te1000

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

